EXHIBIT A

IN THE UNITED STATES DISTRICT COURT
DISTRICT OF DELAWARE

IPA TECHNOLOGIES, INC., Plaintiff, v. AMAZON.COM, INC. and AMAZON DIGITAL SERVICES LLC, Defendants.	C.A. No. 1:16-CV-01266-RGA-SRF
IPA TECHNOLOGIES, INC., Plaintiff, v. GOOGLE LLC, Defendant.	C.A. No. 1:18-CV-00318-RGA-SRF
IPA TECHNOLOGIES, INC., Plaintiff, v. MICROSOFT CORPORATION, Defendant.	C.A. No. 1:18-CV-00001-RGA-SRF

DEFENDANTS' PRELIMINARY INVALIDITY CONTENTIONS

System/Service	Relevant Dates	Persons/Entities Involved in Prior Use, Sale, and/or Offers for Sale	Short Cite
		and also offered for sale and/or sold to its customers this system as evidenced at least by the documents identified herein.	
COMTEC	Pre-1999	COMTEC designed, developed, used, advertised, published, and also offered for sale and/or sold to its customers this system as evidenced at least by the documents identified herein.	"COMTEC"
PRODIGY	Pre-1999	DARPA and CMU, designed, developed, used, advertised, published, and also offered for sale and/or sold to its customers this system as evidenced at least by the documents identified herein.	"PRODIGY"

e. Motivation for Combining Identified Combinations of Prior Art

The combinations of references provided in the accompanying prior art reference charts in Exhibits A-1 through A-24 in combination with A-X are exemplary and are not intended to be exhaustive. Additional obviousness combinations of the references identified here are possible, and Defendants may rely on such combination(s) in this litigation. In particular, Defendants are currently unaware of IPA's allegations with respect to the level of skill in the art and the qualifications of the typical person of ordinary skill in the art. Defendants are also unaware of the extent, if any, to which IPA may contend that limitations of the claims at issue are not disclosed in

been motivated to combine any of the OAA Prior Art. All of these references disclose the use of facilitator agents, user interface agents, and service-providing agents that communicate using an Interagent Communication Language (ICL) to solve user requests, which may take the form of compound goal expressions that are divided into sub-goals and delegated to appropriate agents. *See, e.g.*, OAA Tutorial at 1 (identifying "Open Agent Architecture (OAA) is a multi-agent framework . . . [with a] Facilitator agent [that] coordinates the agent community in achieving the task, providing services such as parallelism, failure handling, conflict detection, and so forth . . . us[ing] [] the Interagent Communication Language [("ICL")]"); OAA Agents at 1 (identifying a few of the "more than 100 agents" that operate within the Open Agent Architecture); PAAM '98 Tutorial at 13 (illustrating agents that communicate with a facilitator using ICL in the OAA Architecture); PAAM '98 Tutorial at 16 and 71 (identifying OAA implementations including Automated Office, Unified Message, Multimodal Maps, Agent Development Tools, MVIEWS, among many others); Cohen at 1 (identifying the "Open Agent Architecture [a]s a blackboard-based framework allowing individual software 'client' agents to communicate by means of goals"); Multimodal Maps (Paper) at 7 (identifying "Open Agent Architecture" as "a framework for coordinating a society of agents . . . [using] [a]n Interagent Communication Language"). A person having ordinary skill in the art would naturally look to the teachings of any OAA Prior Art to solve problems and combine proposed solutions. SRI International also educated the persons of skill in the art about the design, development, functionality, and implementations of its Open Agent Architecture through numerous publications and demonstrations between 1994 and 1997. *See, e.g.*, Cohen (paper dated 1994); Adam Cheyer Demos, *available at* http://www.adam.cheyer.com/demos.html (identifying numerous open agent architecture demonstrations between 1994 and 1997); OAA Tutorial at PAAM-98, Vol 2, no. 1 (January 1,

1998), *available at* http://www.ai.sri.com/~oaa/news/news-v2.1.html (publicly disclosing and presenting a presentation on the details of the Open Agent Architecture at the Third International Conference on the Practical Application of Intelligent Agents held in London in in 1998); Adam Cheyer, FIPA Report (July 1, 1996), *available at* http://www.ai.sri.com/~oaa/distribution/MailArchive/oaa-users/0003.html (confirming that Adam Cheyer attended the 2nd FIPA (Foundation for Intelligent Physical Agents) conference in Yorktown, New York in 1996 and publicly disclosed details of SRI's Open Agent Architecture). Additionally, many of the OAA Prior Art references identify the same authors. A person having ordinary skill in the art would naturally consider other papers, publications, demonstrations, and systems prepared by the same authors.

The asserted patents are also invalid in view of any combination of references related to the KQML and KIF software agent architecture, including, for example, Singh, Genesereth '97, Genesereth '94, Finin I, Finin II, Labrou, InfoSleuth, RETSINA, and any other application or implementation of KQML and KIF (collectively "KQML/KIF Prior Art"). A person having ordinary skill in the art would have been motivated to combine any of the KQML/KIF Prior Art. These references disclose the use of the same Agent Communication Language ("ACL") that uses KQML (Knowledge Query and Manipulation Language) as the "outer" language and KIF (Kowledge Interchange Format) as the "inner" language or vocabulary. *See, e.g.*, Genesereth '97 at 322; Genesereth '94 at 49; Singh at 347-348. The ACL enables one or more facilitators agents to coordinate the activities of connected agents and solve requests by synthesizing a plan that may require decomposing a complex request into a collection of sub-goals and delegating the sub-goals to appropriate agents. *See, e.g.*, Finin II at 1-2; Genesereth '97 at 334; Genesereth '94 at 51; Singh at 339, 345, 354, 356, 358; Labrau at 1. A person having ordinary skill in the art would

naturally look to the teachings of any KQML/KIF Prior Art to solve problems and combine proposed solutions. Indeed, persons of skill in the art maintained publicly available knowledge sharing websites that identified, described, and shared designs and implementations related to the KQML and KIF software architecture. *See, e.g.*, "KQML," University of Maryland, Baltimore County, *available* at https://www.csee.umbc.edu/csee/research/kqml/ (identifying KQML papers, software and applications); "Knowledge Sharing Effort, University of Maryland, Baltimore County, *available at* https://www.csee.umbc.edu/csee/research/kse/ (identifying a ARPA-sponsored knowledge sharing effort related to KQML and KIF); Knowledge Sharing Effort public library, Stanford University, *available at* http://www-ksl.stanford.edu/knowledge-sharing/ (identifying KQML and KIF implementations as part of the ARPA Knowledge Sharing Effort). Additionally, many of the KQML/KIF Prior Art references identify the same authors. A person having ordinary skill in the art would naturally consider other papers, publications, demonstrations, and systems prepared by the same authors.

The asserted patents are also invalid in view of any combination of references related to OAA Prior Art, KQML/KIF Prior Art, FIPA 97, MECCA, Nodine, Bradshaw, Busetta, Nodine, Bayardo, Bian, Malone, Urban, General Magic, White, and any other architecture that implements an agent communication language. A person having ordinary skill in the art would have been motivated to consult and combine the teachings related to these various communication languages to solve problems in another agent communication language. For example, knowledge sharing and collaboration among developers of the various agent communication languages was a central focus of numerous organizations that led international conferences, including, for example, the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM), the Foundation for Intelligent Physical Agents (FIPA), the Association for the Advancement of Artificial

Intelligence (AAAI), and Agent Theories, Architectures, and Languages (ATAL), among others. *See, e.g.*, OAA 2.0 Tutorial at PAAM-98, *available at* http://www.ai.sri.com/~oaa/news/news-v2.1.html; FIPA History, *available* at http://leonardo.chiariglione.org/standards/fipa/fipa_history.htm; ATAL 98, *available at* http://mas.cs.umass.edu/atal/workshops/atal98/index.html; AAAI-97, *available at* http://www.aaai.org/Press/Proceedings/aaai97.php. Participants of these conferences included significant industry player such as IBM, Philips, Siemens, Nokia, Alcatel, France Telecom, Sony, British Telecom, Lucent, and SRI International. *See, e.g.*, FIPA NY Attendance, *available at* http://leonardo.chiariglione.org/standards/fipa/yorktown/nyattendance.htm. These participants routinely shared papers, systems, and other information related to their software agent technology, which often formed the basis for improvements and the development of other standards such as FIPA 1997. *See, e.g.*, Adam Cheyer, FIPA Report (July 1, 1996), *available at* http://www.ai.sri.com/~oaa/distribution/MailArchive/oaa-users/0003.html (Adam Cheyer summarizing various software agent architectures disclosed at the 2nd FIPA (Foundation for Intelligent Physical Agents) conference in Yorktown, New York in 1996); http://leonardo.chiariglione.org/standards/fipa/fipa_history.htm; FIPA Agent Basic Capability List, *available at* http://leonardo.chiariglione.org/standards/fipa/yorktown/nyframework.htm (identifying KQML and KIF as the early foundation for the FIPA specification). Thus, it was common practice for persons having ordinary skill in the art to learn about, consult the teachings of, and implement improvements from the broad array of agent communication languages, architectures, and implementations.

The asserted patents are also invalid in view of any combination of references related to OAA Priori Art, KQML/KIF Prior Art, FIPA 97, MECCA, General Magic and any other agent

communication language that is structured in a similar way. For example, each of these agent communication languages include a conversation layer (specifying events and parameters) and a content layer (specifying goals and/or data) similar to the ICL disclosed by the '115 patent. *See, e.g.,* PAAM '98 Tutorial at 29 (identifying event "oaa_Solve," parameter "query(var(P))," goals "manager() and "phone_number()" and data "John Bear," "M" and "P."); Finin I, slide 49 (identifying event "tell," parameter "ontology ecbk12," goal "price," and data "ISBN3429459,24.95"); FIPA 97, Part 2 (identifying event "inform," parameters "receiver hpl-auction-server" and "ontology hpl-auction," goal "price()" and data "bid good02" and "150"); MECCA at 262 (identifying an event "e," parameters "executing agent," "i-th resource," and "set of constraints on i-th resource," and goal "action to be executed"). A person having ordinary skill in the art would naturally consult and combine teachings of agent communication languages that are structured in a similar way.

The prior art also acknowledged that different agent communication languages could also interoperate. *See, e.g.*, Automated Office at 1:40-1:47 ("This simple interaction style could also be programmed using other distributed technologies available today such as CORBA or Telescript); Genesereth '94 at 50-51 (identifying a "transducer" or "wrapper" that accepts messages from non-native agents and converts those messages into the native agent communication language of the architecture). Thus, a person having ordinary skill in the art would have also been motivated to consult and combine teachings of agent communication languages that may be structured in a different way.

Additionally, the references identified in Section II.A.3., *supra,* all concern the same technological field as the Asserted Patents. In particular, each of the references is directed to the field of software agent architectures and distributed systems. *See, e.g.*, PAAM '98 Tutorial at 4

(SRI presentation describing the Open Agent Architecture, KQML and FIPA as enabling "[c]ooperation among distributed heterogeneous programmatic components."); Kiss at 2:43-2:67 ("Interaction between a user and the knowledge resources is mediated by a collection of cooperative intelligent agents. The cooperative intelligent agents incorporate generalized automated negotiation and distributed inference (i.e., problem-solving) processes."); Malone at Title ("Agents for Information Sharing and Coordination: A History and Some Reflections"); Pollock at 1:39-45 ("integrating the system of defeasible reasoning into an artificial agent that is also capable of planning and executing plans"); Steiner at Abstract ("A Personal Digital Location Apparatus for displaying a geographical location as an icon on a map"). One of ordinary skill in the art would understand these references to all be part of the same field of technology as the Asserted Patents and would naturally look to their teachings to find answers to the problems inherent in the respective technologies. *See, e.g.*, '115 patent, Abstract ("A highly flexible, software-based architecture is disclosed for constructing distributed systems"); 1:22-27 (identifying "Field of the Invention" and the "present invention" as "related to distributed computing environments and the completion of tasks within such environments."); *see also* '128 patent, 1:18-22.

These are only a few examples of the references that one of ordinary skill in the art would consider to be part of the same body of work and in the same technical field and is not meant to be limiting.

Accordingly, a person of ordinary skill in the art at the time of filing of the Asserted Patents would have been motivated to combine elements of any of the references identified here and recognize that the combination of any of these references is a predictable use of elements known in the art to solve a known problem and a use of known techniques to solve a known problem in

the same way. Defendants' expert(s) may further explain the motivation to combine prior art and why the Asserted Claims of the Asserted Patents are invalid for obviousness in accordance with the case schedule.

4. INVALIDITY UNDER § 112

The Asserted Claims are invalid under 35 U.S.C. § 112. The Asserted Claims lack a written description and enabling disclosure commensurate with the alleged scope of the claims, are unduly vague and indefinite, and contain purely functional language. The Asserted Patents, read in light of the specification and prosecution history, fail to inform, with reasonable certainty, those skilled in the art about the scope of the invention. *See Nautilus, Inc. v. Biosig Instruments, Inc.*, 134 S. Ct. 2120, 2124 (2014). The Asserted Patents do not enable one of skill in the art to practice the full scope of the inventions claimed without undue experimentation. The Asserted Patents do not enable one of skill in the art to practice the scope of the inventions set forth in IPA's preliminary infringement contentions.

Because expert disclosures are not yet due, including those in connection with claim construction, Defendants reserve the right to raise any additional issues from the perspective of one skilled in the art at the appropriate time in the case.

The following identification of claims/claim elements are preliminary and only exemplary and Defendants reserve the right to supplement the identification of claims and claim elements that do not comply with the requirements of 35 U.S.C. § 112. Specifically, to the extent an element identified below, or its variation, appears in claims other than the ones specified below, it also renders those additional claims invalid under 35 U.S.C. § 112. Claims that depend on these additional claims and on the claims identified below are also invalid under 35 U.S.C. § 112. Defendants reserve the right to identify additional claims and claim elements that do not comply with the requirements of 35 U.S.C. § 112 during claim construction or after the Court construes

EXHIBIT B

Exhibit C-7

Invalidity of U.S. Patent No. 7,069,560 ("'560 Patent")
by Genesereth '97

As shown in the claim chart below, the asserted claims of the '560 patent are invalid under 35 U.S.C. § 102(a) and/or (b)[1] as anticipated by Michael R. Genesereth, "An Agent-Based Framework of Interoperability" (1997)[2] ("Genesereth '97"), and/or are invalid under 35 U.S.C. § 103 as obvious in view of Genesereth '97, or in combination with the reference(s) specifically identified in the following claim chart or one or more other references identified in Defendants' Preliminary Invalidity Contentions ("Invalidity Contentions").

The additional KIF/KQML documents relied upon herein include, but are not limited to:

- Yannis Labrou, et al., "Semantics for an Agent Communication Language" (1997)[3] ("Labrou"). Labrou provides further descriptions and examples of the types of syntax and expressions possible with the KIF/KQML architecture that is described in Genesereth '97.

To the extent a finder of fact determines that the references cited herein do not teach certain limitations in the asserted claims, such limitations would have been inherent and/or obvious. These claims are also invalid as obvious in view of Genesereth '97 alone or in combination with other prior art references, including, but not limited, to the prior art identified in the Cover Pleading of Defendants' Invalidity Contentions, the prior art described in the claim charts attached in Appendix C, and/or the prior art identified in Appendix D.

Defendants' Invalidity Contentions are based, in part, upon Defendants' present understanding of the asserted claims and IPA's apparent interpretations of the asserted claims in its July 10, 2019 Preliminary Infringement Contentions ("Infringement Contentions") and Defendants' investigation to date. Defendants are not adopting IPA's constructions or apparent constructions, nor are Defendants admitting to the accuracy of any particular contention or construction. The citations provided in the charts below are exemplary rather than exhaustive and Defendants reserve the right to rely upon additional references uncovered through further searching, other portions of the cited references and/or other portions of references cited within these Invalidity Contentions. Defendants further incorporate by reference the reservation of rights identified in the cover pleading to these Invalidity Contentions as though fully set

[1] Genesereth '97 was published in 1997 more than one year before the January 5, 1999 earliest possible filing date of the '560 patent.
[2] Published in Jeffrey M. Bradshaw, "Software Agents," American Association for Artificial Intelligence (1997).
[3] Published in Munindar P. Singh, et al., Intelligent Agents IV Agent Theories, Architectures, and Languages, ATAL '97, vol 1365.

forth herein.

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(p)	A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of	To the extent the preamble is found to be limiting, Genesereth ’97 discloses a computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of claim 1. *See* Claim Chart for U.S. Patent No. 6,851,115 in view of Genesereth ’97, Ex. A-7 (“’115 chart”), claim 1.
1(a)	a plurality of service-providing electronic agents;	Genesereth ’97 discloses a plurality of service-providing electronic agents. *See* ’115 chart, claim 1.
1(b)	a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:	Genesereth ’97 discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,* Genesereth ’97 at 320 (bold emphasis added). > Facilitators and the agents they manage are typically organized into what is often called a *federated system*. Figure 2 illustrates the structure of such a system in the simple case in which there are just three machines, one with three agents and two with two agents apiece. As suggested by the diagram, agents do not communicate directly with each other. Instead, they communicate only with their **local facilitators**, and **facilitators, in turn, communicate with each other**. In effect, the agents form a “federation” in which they surrender their autonomy to the facilitators. *See also,* Genesereth ’97 at Fig. 2.

	’560 Patent Claim Language	Invalidity in View of Prior Art

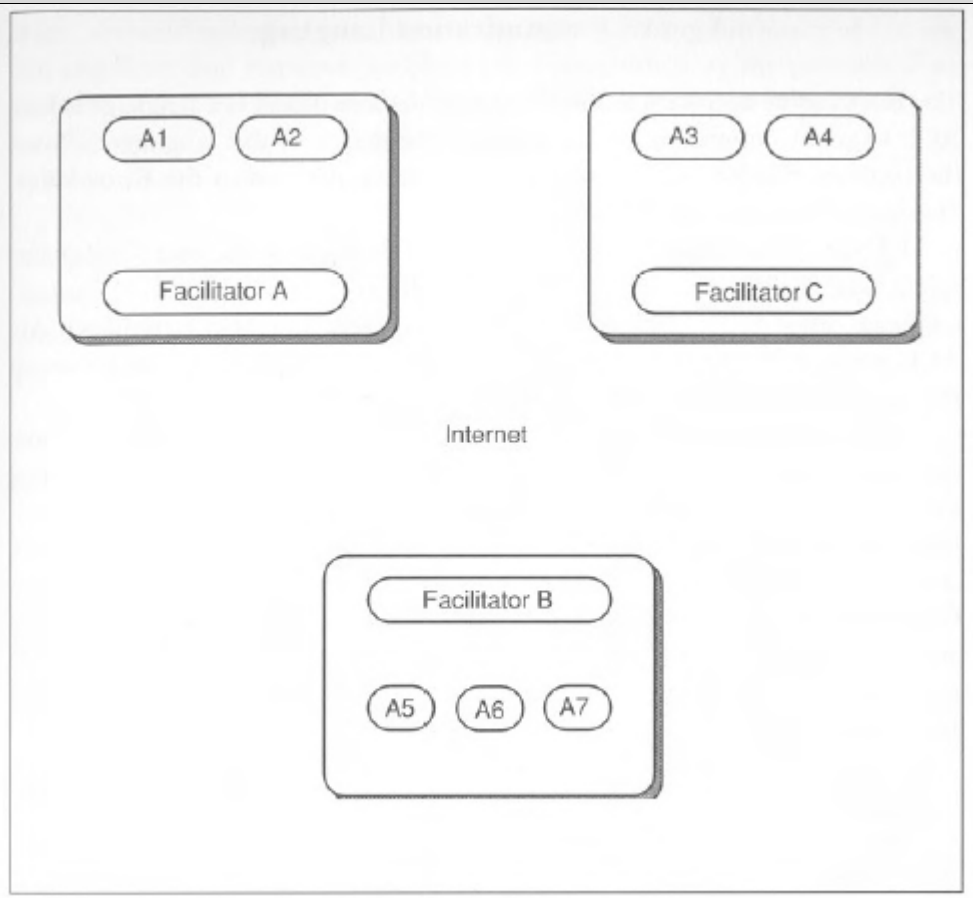


Figure 2. Federated system.

See also, Genesereth ’97 at 320 (emphasis added).

> As with most other brokering approaches, messages from servers to facilitators are undirected; i.e., they have content but no addresses. It is the responsibility of the **facilitators to route such messages to agents able to handle them**. There can be an arbitrary number of facilitators, on one or more machines, and

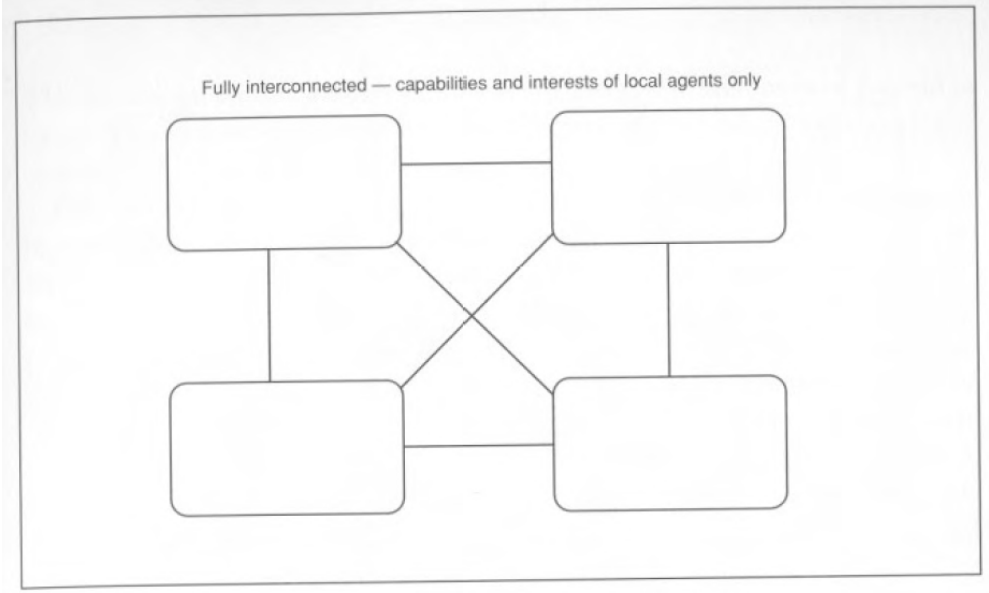
	’560 Patent Claim Language	Invalidity in View of Prior Art
		the **network of facilitators** can be connected arbitrarily. *See also,* Genesereth ’97 at 320 (bold emphasis added). The federation architecture provides assisted coordination of other agents based on a *specification-sharing* approach to interoperation. Agents can dynamically connect or disconnect from a facilitator. **Upon connecting to a facilitator, an agent supplies a specification of its *capabilities* and *needs* in ACL. In addition to this meta-level information, agents also send application-level information and requests to their facilitators and accept application-level information and requests in return. Facilitators used the documentation provided by these agents to transform these application-level messages and route them to the appropriate agents.** The agents agree to service the requests sent by the facilitators, and in return, the facilitators manage the requests posted by the agents. *See also,* Genesereth ’97 at 320-21. A major difference between the knowledge-sharing approach to software interoperation and previous approaches lies in the sophistication of the processing done by these facilitators. In some cases, facilitators may have to translate the messages from the sender's form into a form acceptable to the recipient. In some cases, they may have to decompose the message into several messages, sent to different agents. In some cases, they may combine multiple messages. In some cases, this assistance can be rendered interpretively, with messages going through the facilitators; in other cases, it can be done in one-shot fashion, with the facilitators setting up specialized links between individual agents and then stepping out of the picture. *See also,* Genesereth ’97 at 325-26 (emphasis added). In the approach to interoperation described here, application programmers write their programs as software agents. Like other agents, **a software agent is obliged to communicate in ACL**, but it does so in a particularly stylized way:

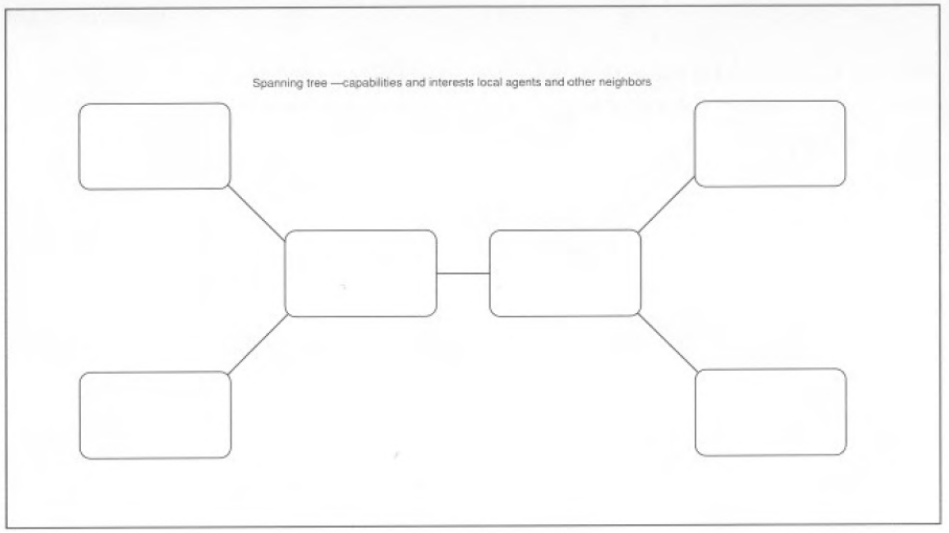
	’560 Patent Claim Language	Invalidity in View of Prior Art
		l. On start up, it initiates an ACL connection to the local facilitator. **2. It supplies the facilitator with a description of its capabilities.** 3. It then enters normal operation: **it sends the facilitator requests when it is incapable of fulfilling its own needs, and it acts to the best of its abilities to satisfy the facilitator's requests**. A software agent is a special kind of agent in that it surrenders its autonomy to the facilitator. A general agent is not compelled to satisfy the requests of other agents. It can accept them or decline them, or it can negotiate for payment. A software agent does not have this freedom. **After registering with its local facilitator and supplying its specification, the software agent is obliged to satisfy the facilitator's requests whenever it can. Of course, this is a good deal in many cases, since the agent gets the facilitator's services in return**. *See also,* Genesereth ’97 at 326 (emphasis added). **Specifying Agent Capabilities and Needs** In order to provide services to other agents, **an agent must communicate its capabilities to the facilitator in ACL. An agent specifies its capabilities by transmitting “handles” facts to its facilitator.** For example, an agent capable of answering questions about the dealer of a vendor may transmit the following specification to its facilitator: (handles business-agent‘(ask-one ,?variables(dealer ,?dealer ,?vendor))) (handles business-agent‘(ask-all ,?variables(dealer ,?dealer ,?vendor))) These facts state that agent business-agent is capable of answering queries about a single dealer for a vendor, or all the dealers for a vendor. The actual capability is a quoted KQML expression, such as ‘(ask-one ,?variables(dealer ,?dealer ,?vendor)) in the first example. This specification is similar to the object interface specifications in CORBA’s IDL.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		If some other agent A_1 wants to know the dealers of NEC, it may communicate the following request to the facilitator: (ask-all ?x (dealer ?x nec)) The facilitator examines its knowledge base and determines that the business-agent can handle the request. **The facilitator sends the request to the business-agent, gets the answer, and passes it to A_1**. Agent A_1 is completely unaware of the sequence of steps performed in servicing its request. *See also,* Genesereth ’97 at 326-27 (emphasis added). **An agent specifies its needs by transmitting “interested” facts to its facilitator.** For example, the following states that the agent cs-manager is interested in all facts regarding the release of PC-compatible computers. (interested cs-manager ‘(tell (released ,?manufacturer PC ,?model))) Similar to “handles” statements, “interested” statements can be conditional: (<= (interested cs-manager ‘(tell (released ,?manufacturer PC ,?model))) (member ?manufacturer ‘(ibm toshiba nec micro-international))) This states that the cs-manager agent is interested only in the release of PC-compatible computers from IBM, Toshiba, NEC, and Micro-International. **If another agent transmits the following fact to the facilitator:** **(tell (released micro-international PC 6500D))** **then the facilitator will examine its knowledge base and find that the agent cs-manager is interested in expressions of this form, and it will send the same KQML expression to the cs-manager**. *See also,* Genesereth ’97 at 329 (emphasis added). Facilitators are the system-provided agents that coordinate the activities of the other agents in the federation architecture. **Each facilitator keeps the other facilitators in the network informed of which agents are connected to it and**

	’560 Patent Claim Language	Invalidity in View of Prior Art
		what facts have been communicated by them. *See also,* Genesereth ’97 at 329 (bold emphasis added). Facilitators provide a collection of services, including: • *White Pages:* finding the identity of agents by name, for example, “What agents are connected?” or “Is agent x connected?” • *Yellow Pages:* finding the identity of agents capable of performing a task. For example, “What agents are capable of answering the query x?” • *Direct Communication:* sending a message to a specific agent. • *Content-based Routing:* the facilitator is given the responsibility of handling a request. It makes use of the specifications and other information provided by the agents to do this, thereby giving the illusion that it is the sole provider of all services. • *Translation:* agents may use different vocabulary. In order to interoperate, the facilitator may have to translate the vocabulary of one agent into the vocabulary of another. • *Problem Decomposition:* handling a complex request may require breaking it into sub-problems, getting the answers to the sub-problems, and then combining these answers to obtain the answer to the original request. As in content-based routing, the facilitator makes use of the specifications and application-specific information provided by the agents to accomplish this. • *Monitoring:* when an agent informs the facilitator of a need, the facilitator monitors its knowledge to determine if the need can be satisfied. For example, an agent may specify the need “I am interested in facts about the position of chips in design x.” **The responsibility of the facilitator on each machine is to assist the agents running on that machine to collaborate with each other and, indirectly, with the agents running on other machines.** *See also,* Genesereth ’97 at 336-38 (disclosing multiple facilitators on same machine each on a different processor and connection of remote and local agents and facilitators)

	’560 Patent Claim Language	Invalidity in View of Prior Art
		(emphasis added). Since remote communication is more expensive than local communication, there is good reason for having at least one facilitator on each machine. Otherwise, in order for a program to communicate with another program on the *same* machine, it would have to send a message to a *remote* machine! On the other hand, there is really no reason to have more than one facilitator per machine. Anything that can be handled by two facilitators can be handled by one facilitator. **There can be no computational advantage, unless the two facilitators are running on different processors with the same machine**. What about the connection of agents to facilitators? While it is possible to consider a situation in which every agent is connected to every facilitator, this is impractical in settings, like the Internet, where there are likely to be many agents and many facilitators. For this reason, in federation architecture, I assume that every agent is connected to one and only one facilitator Finally, there is the issue of inter-facilitator connectivity. Here, there are multiple choices, each with advantages and disadvantages. The simplest sort of architecture is full interconnection, as suggested by figure 4. In this architecture, every facilitator is connected to every other facilitator. Since these connections are logical connections and not physical wires, this sort of architecture is feasible, though not necessarily desirable.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 *Figure 4. Full interconnection architecture.* * * * An alternative that alleviates this difficulty is a spanning tree architecture, as suggested in figure 5. In this approach, facilitators are connected in such a way that there is a path from every facilitator to every other facilitator but there are no loops.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 *Figure 5. Spanning tree architecture.* * * * Finally, there is the general connectivity architecture. In this architecture, every facilitator is connected at least indirectly with every other facilitator, as in the spanning tree architecture, but there is no restriction that the connectivity be loop free. * * * Unfortunately, it has the disadvantage of possible loops. If one facilitator sends a message to a second and the second passes it on to a third and the third passes it on again, it might end up back where it started. Fortunately, loops of this sort can be caught by adding sender information to each message (as in many mail protocols, for example) and checking for this information when a message is received. It can also be handled by having each

	’560 Patent Claim Language	Invalidity in View of Prior Art
		facilitator save information about which messages it has sent. Either way the loops can be broken. The programming cost is a little higher, but the efficiency and reliability of the approach recommend it highly. Another complexity in the spanning tree and general connectivity architectures stems from the need of facilitators to merge the interests of other facilitators in with those of their own agents in complicated ways. In a full connectivity architecture, each facilitator simply aggregates the interests of its local agents and passes those interests on to all other facilitators. Each facilitator uses this information to handle incoming requests. In the other two architectures, the interests passed on to neighbors are more complicated. A facilitator connected to two other facilitators must blend the interests of its first neighbor into the interests of its local agents in the specification it sends to its second neighbor; and it must blend the interests of its second neighbor into the interests of its local agents in the specification it sends to its first neighbor. *See also,* Genesereth ’97 at 339 (emphasis added). This section presents a simple example of the **federation architecture**. Instead of focusing on the details, I present a broad picture of the types of software interoperation made possible. First, a brief overview of the scenario. There is a computer systems manager in a publishing company who wants to upgrade the computers used by the sales staff to portable Pentium-based machines. The computer systems manager **informs the facilitator of his interest** in Pentium laptops. Sometime later, the **computer product agent notifies the facilitator of the availability** of a Pentium laptop, and this information is passed on to the computer systems manager by the facilitator. The computer systems manager **asks the meeting scheduling agent to set up a joint meeting** with the managers of the sales and finance departments to discuss the purchase of the new machines. **The meeting scheduling agent gets the available times from the calendar agents** for the sales and finance managers to schedule a meeting.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		See also, Genesereth ’97 at 341. Finally, the example also illustrates the dual nature of agents as both providers and consumers of services. For example, the meeting scheduling agent can handle a request to schedule a meeting. However, in order to service this request, the scheduling agent must ask the facilitator for the calendars of the participants. *See also,* Genesereth ’97 at 343-44 (emphasis added). Similarly, it is not possible to put a bound on the total number of agents in the system. **A system can have a network of facilitators, with different agents connected to different facilitators.** Each facilitator must be capable of transmitting a request to any agent that can handle it, independent of its location. To minimize the number of capability and interest specification facts, each facilitator summarizes the capabilities and interests of its directly connected agents, and passes on this summary to its neighboring facilitators. The summary reduces the number of facts and may involve generalization. For example, if one directly-connected agent can answer questions about the dealers of Apple computers and another directly-connected agent can answer questions about IBM dealers, then the facilitator may summarize the answers by informing its neighboring facilitators that it can answer questions about the dealers of all personal computers. There is a space-time tradeoff here: fewer less-precise specifications vs. a larger number of more precise specifications. It is acceptable for an agent to handle a request by indicating that it cannot answer it, for example if its specifications are too general.
1(c)	an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment; and	Genesereth ’97 discloses an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment. *See* ’115 chart, claim 1(a).

	'560 Patent Claim Language	Invalidity in View of Prior Art
1(d)	a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves:	Genesereth '97 discloses a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves. *See* '115 chart, claims 1(e), (h).
1(e)	using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal; and	Genesereth '97 discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* '115 chart, claim 1(g)-(i). *See also, e.g.*, Genesereth '97 at 329 (bold emphasis added). Facilitators provide a collection of services, including: • *White Pages:* finding the identity of agents by name, for example, "What agents are connected?" or "Is agent x connected?" • *Yellow Pages:* finding the identity of agents capable of performing a task. For example, "What agents are capable of answering the query x?" • *Direct Communication:* sending a message to a specific agent. • *Content-based Routing:* the facilitator is given the responsibility of handling a request. It makes use of the specifications and other information provided by the agents to do this, thereby giving the illusion that it is the sole provider of all services. • *Translation:* agents may use different vocabulary. In order to interoperate, the facilitator may have to translate the vocabulary of one agent into the vocabulary of another. • ***Problem Decomposition:* handling a complex request may require breaking it into sub-problems, getting the answers to the sub-problems, and then combining these answers to obtain the answer to the original request. As in content-based routing, the facilitator makes use of the**

	’560 Patent Claim Language	Invalidity in View of Prior Art
		specifications and application-specific information provided by the agents to accomplish this. • *Monitoring:* when an agent informs the facilitator of a need, the facilitator monitors its knowledge to determine if the need can be satisfied. For example, an agent may specify the need “I am interested in facts about the position of chips in design x.” The responsibility of the facilitator on each machine is to assist the agents running on that machine to collaborate with each other and, indirectly, with the agents running on other machines. *See also,* Genesereth ’97 at 329-30 (emphasis added). In handling a message, the message handler uses an **automated reasoning program on its knowledge base of specification information**. Our reasoning program is a variation on the method used in Prolog. There are two primary differences. First of all, it handles KIF syntax rather than Prolog syntax. Secondly, unlike Prolog, it is sound and complete for full first-order predicate calculus: it is based on the model elimination rule of inference, the unification algorithm does an occurcheck, the restriction to Horn clauses is removed, and the search is done in iterative deepening fashion. *See also,* Genesereth ’97 at 330-31. Consider a database with the sentences shown below. The predicate p holds of three pairs of objects-a and a, a and b, and b and e. The predicate q is also true of three pairs of objects-a and b, b and e, and e and d. The predicate r is defined to be true of two objects if there is an intermediate object such that p is true of the former object and this intermediate object and q is true of the intermediate object and the latter object. (p a a) (p a b) (p b c)

	’560 Patent Claim Language	Invalidity in View of Prior Art
		(q a b) (q b c) (q c d) (<= (r ?x ?z) (p ?x ?y) (q ?y ?z)) Suppose now, we wanted to know whether r was true of a and c. The trace shown below shows how the reasoning program derives this result. Call: (r a c)? Call: (and (p a ?y| (q ?y c)) Call: (p a ?y) Exit: (p a a) Call: (q a c) Fail: (q a c) Call: (p a ?y) Exit: (p a b) Call: (q b c) Exit: (q b c) Call: (and (p a b) |q b c)) Exit:(r a c) The desired conclusion (r a c) unifies with the conclusion of the last sentence in the knowledge base with the variable ?x bound to a and the variable ?z bound to c. The program thus reduces the original question to the subquestion on the

	’560 Patent Claim Language	Invalidity in View of Prior Art
		second line-in effect the question of whether there is a binding for the variable ?y for which the conjunction is true. In order for this to be true, there must be a binding for ?y for which (p a ?y) is true. The program first finds (p a a) and binds ?y to a. It then tries to prove (q a c). Unfortunately, this fails. So, the program backs up and tries to find another way to satisfy (p a ?y). In so doing, it discovers the fact (p a b) and binds ?y to b. Again it tries to prove (q b c) and in this case succeeds. Since both conjuncts are proved, the conjunction is proved; and, since the conjunction is proved, the original sentence is proved. *See also,* Genesereth ’97 at 331. This program is both sound and complete. In other words, if the program manages to prove a result, then that result must logically follow from the sentences in the database; and if a conclusion logically follows from the database, the method will prove it. Unfortunately, as with all sound and complete reasoning methods for the full first-order predicate calculus, the method does not necessarily terminate. If a conclusion does not follow from the database, the method may spend forever trying to prove it. While this situation does not often arise, it is a real danger for a piece of system code. In order to deal with this difficulty, the facilitator uses a preprocessor to screen sentences before they are added to the facilitator's database. The facilitator adds a sentence if and only if it can prove that doing so will not cause an infinite loop. Note that the problem of making this determination is itself undecidable; so it is not possible to know in all cases whether a sentence will cause an infinite loop. Our facilitator circumvents this difficulty by taking a conservative approach to proving the “safeness” of a set of sentences: it uses a variety of tests to determine whether an inference will terminate. If a database passes the tests, termination is assured. If not, the database may or may not be safe. *See also,* Genesereth ’97 at 332.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		Content-Based Routing From an application programmer’s point of view, communication in a federation architecture is undirected: application programs are free to send messages without specifying destinations for those messages. It is the job of the facilitator to determine appropriate recipients for undirected messages and to forward the messages accordingly. In so doing, the facilitator functions as a broker for the services provided by the servers in its community. In order to see how this is done, consider how the facilitator handles the message shown below. It is being told via one particular encoding that the object named chip1 is indeed a computer chip. (tell '(member chip1 chips)) The facilitator is connected to three agents, named layout, domain-editor, and board-editor. These agents have given the facilitator the specification information shown below. (interested layout '(tell (position ,?x ,?r ,?c))) (<= (interested domain-editor '(tell (member ,?x ,?y))) (symbol ?x) (symbol ?y)) (<= (interested board-editor '(tell (= (,?f ,?x) ,?y))) (member ?f (setoff 'row 'col)) (symbol ?x) (natural-number ?y)) In order to determine which agents are interested in this message, the facilitator forms the query (interested ?a '(tell (member chip1 chips))) and uses its reasoning program to find a binding for variable ?a. In this case, there is just

	’560 Patent Claim Language	Invalidity in View of Prior Art
		one, the domain-editor. Consequently, the facilitator sends the message to this agent. (tell '(member chip1 chips)) Note that in making the determination that the domain-editor agent is interested, the facilitator must not only match the pattern in the first line of its specification but also verify properties of the bindings of the variables, in particular that they are both symbols. *See also,* Genesereth ’97 at 333 (emphasis added). As an example of translation, consider a situation in which the facilitator receives the message shown below. As before, it is being told via one particular encoding that the position of a particular chip on a printed circuit board is located in the tenth row and sixteenth column. (tell '(= (pos chip1) (point 10 16))) The facilitator’s agent catalog mentions that an agent named layout is interested in receiving messages of the form (position **), where ** and ** are natural numbers. (<= (interested kb (tell '(position ?x ?m ?n)) (natural-number ?m) (natural-number ?n)) Since the incoming sentence does not have the form specified in this interest, content-based routing alone would not cause any message to be sent to layout. However, let us suppose that the facilitator also has information relating pos and position, as in the following sentence: (<=> (= (pos ?x) (point ?row ?col)) (position ?x ?row ?col)) **Using this sentence together with the sentence (= (pos chip1) (point 10 16)),**

	’560 Patent Claim Language	Invalidity in View of Prior Art
		the facilitator is able to deduce the sentence (position chip1 10 16). In other words, it can translate from one form to the other. It then checks whether any agent is interested in this information, finds layout, and sends the message shown below. (tell '(position chip1 10 16)) *See also,* Genesereth ’97 at 334-35 (emphasis added). The example of translation in the preceding subsection is particularly simple. One incoming message leads to one outgoing message. **In some cases, an incoming message can be handled only by sending multiple messages to multiple agents. In order to handle such messages, the facilitator must be able to synthesize a multi-step communication plan to handle the incoming message**. As an example of this type of message handling, consider the message shown below. As in the last example, the facilitator is being told the position of a particular chip. (tell '(= (pos chip1) (point 10 16))) One difference in this example is that the facilitator’s agent catalog contains the information shown below, documenting an agent interested in row information and col information but not pos information. (<= (interested board-editor (tell '(= (,?f ,?x) ,?y))) (member ?f (setof 'row 'col)) (symbol ?x) (natural-number ?y)) As before, let us assume that the facilitator’s library contains a sentence relating the two vocabularies. (<=> (= (pos ?x) (point ?row ?col))

	’560 Patent Claim Language	Invalidity in View of Prior Art
		(and (= (row ?x) ?row) (= (col ?x) ?col)))
		In this case, there are two conclusions from the original sentence. The facilitator discovers these two conclusions and sends them on to the board-editor agent.
		(tell '(= (row chipl) 100))
		(tell '(= (col chip1) 160))
		Note that if the incoming message had been an ask-if message, the facilitator would have been able to reduce this to two questions: one about the row of the chip and another about the column. In this case, it would first send the row question to board-editor; then, on getting an answer, it would send in the col question; and, on getting that answer, would be able to answer the original question.
		See also, Genesereth ’97 at 339-40.
		The computer systems manager sits at his terminal with a graphical user interface (GUI) and tells the facilitator that he is interested in being told of the availability of PC-compatible Pentium laptops. The GUI commands are translated into the following KIF fact, which is transmitted to the facilitator:
		(<= interested cs-manager '(tell (available ,?manufacturer ,?model-name)))
		(= (denotation ?model-name) ?model); the model from its name
		(computer-family ?model PC)
		(laptop ?model))
		There is a product agent that can answer queries about the computer family a product belongs to (e.g., PC, Apple) and which computers are laptops. It has specified its capabilities by transmitting the following facts to the facilitator:
		(handles product-agent
		'(ask-one ,?variables (computer-family ,?computer ,?family)))

	’560 Patent Claim Language	Invalidity in View of Prior Art
		(handles product-agent '(ask-if (laptop ,?computer))) Whenever a new piece of information is added to the product agent's knowledge base it notifies the facilitator of the fact. A new Micro-International 36000 computer is announced, and information about it is added to the knowledge base of the product agent. The product agent communicates the following KQML message to the facilitator: (tell (available Micro-International 3600D)) The facilitator performs inference to see if any agent is interested in this fact. It finds that the cs-manager agent is interested, but only if the computer family of the 36000 is PC, and if the 36000 is a laptop. The facilitator cannot answer these questions locally. However, it forwards the queries to the product-agent, who can answer them. The product-agent responds positively to both queries, and the cs-manager is notified of the previous availability of the 36000. A message indicating this pops up on the GUI of the computer systems manager. The computer systems manager uses his GUI to ask the facilitator to schedule a one hour meeting with the managers of the sales and finance groups during the week of December 12th to 16th. The GUI transmits the following KQML message to the facilitator: (schedule-meeting (listof sales-manager finance-manager) (interval 12-12-94 12-16-94) 60) There is a scheduling agent that can schedule meetings. It previously transmitted the following fact to the facilitator: (handles scheduler '(schedule-meeting ,?people ,?interval ,?meeting-duration)) The original meeting request is passed on to the scheduler agent by the facilitator. The scheduler is not able to schedule a meeting directly, since it does

	’560 Patent Claim Language	Invalidity in View of Prior Art
		not have access to the calendars of the sales and finance managers. Therefore, the scheduling agent passes on the following query to the facilitator: (ask-one ?x (calendar sales-manager (interval 12-12-94 12-16-94) ?x)) There is a datebook agent for the sales manager that records his calendar. It had previously notified the facilitator of its capability with the following fact: (handles sales-manager-datebook '(ask-one ,?x (calendar sales-manager ,?interval ,?x))) Similarly, there is a synchronize agent that can answer queries regarding the calendar of the finance manager. The facilitator passes on the two queries of the scheduler to the sales-manager-datebook agent and the finance-manager-synchronize agent. The calendars returned by these agents are sent to the scheduling agent, who schedules the earliest possible meeting. The first available meeting time is transmitted to the facilitator, who finally forwards the results to the cs-manager. Genesereth ’97 discloses this limitation as identified above. This limitation is also obvious in view of Genesereth ’97 combined with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X. As Ex. C-X shows, each of these references also discloses this limitation. A person having ordinary skill in the art would have been motivated to combine Genesereth ’97 with one or more of the references identified in the corresponding limitation of Ex. C-X rendering this limitation obvious based on one or more of the motivations to combine identified in Section II.A.3.e of the cover pleading to Defendants’ Preliminary Invalidity Contentions and because each of these references relate to the same field of software agent technology and distributed computing environments. Additionally, this limitation is also disclosed by Labrou. A person having ordinary skill in the art would have been motivated to combine Genesereth ’97 with Labrou based on one or more of the motivations to combine identified in Section II.A.3.e of the cover

	’560 Patent Claim Language	Invalidity in View of Prior Art
		pleading to Defendants’ Preliminary Invalidity Contentions. These references both relate to the KQML/KIF architecture and Labrou identifies information about the semantics of KQML such as the use of conditional expressions, conjunctive and disjunctive operators, and parameters. *See also*, Labrou at 1. A crucial component of this paradigm is the communication language, which is the medium through which the attitudes regarding the content of an exchange between software agents are communicated; the communication language suggests whether the content of the communication is an assertion, a request, some form of query *etc.* Knowledge Query and Manipulation Language (KQML) is an agent communication language that consists of primitives (called *performatives*) which allow agents to communicate such attitudes to other agents and find other agents suitable to process their requests. Our research provides semantics for KQML along with a framework for the semantic description of KQML-like languages for agent communication. We do so, avoiding commitments to agent models and inter-agent interaction protocols. *See also*, Labrou at 2. This is an example of a KQML message: `(ask-if :sender A :receiver B :language prolog :ontology foo :reply-with id1 :content ‘‘bar(a,b)’’ )` In KQML terminology, *ask-if* is a *performative*. The value of the :content is an expression in some language (in this case in Prolog) or another KQML message and represents the content of the communication (illocutionary) act. The other parameters (*keywords*) introduce values that provide a context for the interpretation of the :content and hold information to facilitate the processing of the message. *See also*, Labrou at 3.1. The following constitutes the semantic description for each of the performatives: **(1)** A natural language description of the performative’s intuitive

	’560 Patent Claim Language	Invalidity in View of Prior Art
		meaning; **(2)** An expression that describes the content of the communication act and serves as a formalization of the natural language description; **(3)** Preconditions that indicate the necessary state for an agent in order to send a performative (**Pre(A)**) and for the receiver to accept it and successfully process it (**Pre(B)**); **(4)** Postconditions that describe the states of both interlocutors after the *successful* utterance of a performative (by the sender) and after the receipt and processing (but before a counter utterance) of a message (by the receiver). The postconditions (**Post(A)** and **Post(B)**, respectively) hold unless a *sorry* or an *error* is sent as a *response* in order to suggest the unsuccessful processing of the message; **(5)** A completion condition for the performative (**Completion**) that indicates the final state, after possibly a conversation has taken place and the intention suggested by the performative that started the conversation, has been fulfilled; and **(6)** Any comments that we might find suitable to enhance the understanding of the performative. *See also*, Labrou at 3.3. For a KQML message **performative(A,B,X)**, **A** is the :sender, **B** is the :receiver and **X** is the :content of the performative (KQML message). * * * All expressions in our language denote agents’ states. Agents’ states are either actions that have occurred (PROC and SENDMSG) or agents’ mental states (BEL, KNOW, WANT or INT). We allow conjunctions (∧) and disjunctions (∨) of expressions that stand for agents’ states (the resulting expressions represent agents’ states, also), but we do not allow ∧ and ∨ in the scope of KNOW, WANT and INT. *See also*, Labrou at 4.

	'560 Patent Claim Language	Invalidity in View of Prior Art
		We present the semantics for three KQML performatives (*ask-if*, *tell* and *sorry*) in order to illustrate our approach. [1] – **ask-if(A,B,X)** 1. A wants to know what B believes regarding the truth status of the content X. 2. WANT(A,KNOW(A,S)) where S may be any of BEL(B,X), or ¬(BEL(B,X)). 3. **Pre(A)**: WANT(A,KNOW(A,S)) ∧ KNOW(A,INT(B,PROC(B,M))) where M is **ask-if(A,B,X)** **Pre(B)**: INT(B,PROC(B,M)) 4. **Post(A)**: INT(A,KNOW(A,S)) **Post(B)**: KNOW(B,WANT(A,KNOW(A,S))) 5. **Completion**: KNOW(A,S')) where S' is either BEL(B,X) or ¬(BEL(B,X)), but not necessarily the same instantiation of S that appears in $Post(A)$, for example. 6. Not believing something is not necessarily the same with believing its negation (assuming that the language of B provides logical negation), although this may be the case for certain systems. The **Pre(A)** and **Pre(B)** suggest that a proper advertisement is needed to establish them. – **tell(A,B,X)** 1. A states to B that A believes the content to be true. 2. BEL(A,X) 3. **Pre(A)**: BEL(A,X) ∧ KNOW(A,WANT(B,KNOW(B,S))) **Pre(B)**: INT(B,KNOW(B,S)) where S may be any of BEL(B,X), or ¬(BEL(B,X)).

	’560 Patent Claim Language	Invalidity in View of Prior Art
		4. **Post(A)**: KNOW(A,KNOW(B,BEL(A,X))) **Post(B)**: KNOW(B,BEL(A,X)) 5. **Completion**: KNOW(B,BEL(A,X)) 6. The completion condition holds, unless a *sorry* or *error* suggests B’s inability to acknowledge the *tell* properly, as is the case with any other performative. – **sorry(A,B,Id)** 1. A states to B that although it processed the message, it has no response to provide to the KQML message M identified by the `:reply-with` value **Id** (some message identifier). 2. PROC(A,M) 3. **Pre(A)**: PROC(A,M) **Pre(B)**: SENDMSG(B,A,M) 4. **Post(A)**: KNOW(A,KNOW(B,PROC(A,M))) $\wedge$ not$(Post_M(A))$, where $Post_M(A)$ is the **Post(A)** for message M. **Post(B)**: KNOW(B,PROC(A,M)) $\wedge$ not$(Post_M(B))$ 5. **Completion**: KNOW(B,PROC(A,M)) 6. The postconditions for M, as a result of message M do not hold. The not should be taken to mean that the mental state it qualifies should not be inferred to be true as a *result* of this particular message. This does not mean that for example $Post_M(B)$ does not hold if it has already been established by a previous message; it is up to B to decide (perhaps after using additional information) if and how it wants to alter its internal state with respect to the *sorry*.
1(f)	wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes:	Genesereth ’97 discloses wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes. *See* ’115 chart, claims 1(a)-(c).

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(g)	a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events.	Genesereth ’97 discloses a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events. *See* ’115 chart, claims 1(b)-(c).
20	A computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process.	Genesereth ’97 discloses a computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process. *See, e.g.*, Genesereth ’97 at 336-38 (disclosing multiple facilitators on same machine each on a different processor and connection of remote facilitators) (bold emphasis added). Since remote communication is more expensive than local communication, there is good reason for having at least one facilitator on each machine. Otherwise, in order for a program to communicate with another program on the *same* machine, it would have to send a message to a *remote* machine! On the other hand, there is really no reason to have more than one facilitator per machine. Anything that can be handled by two facilitators can be handled by one facilitator. There can be no computational advantage, **unless the two facilitators are running on different processors with the same machine**. What about the connection of agents to facilitators? While it is possible to consider a situation in which every agent is connected to every facilitator, this is impractical in settings, like the Internet, where there are likely to be many agents and many facilitators. For this reason, in federation architecture, I assume that every agent is connected to one and only one facilitator Finally, there is the issue of inter-facilitator connectivity. Here, there are multiple choices, each with advantages and disadvantages.

EXHIBIT C

Exhibit C-8

Invalidity of U.S. Patent No. 7,069,560 ("'560 Patent")
by Singh

As shown in the claim chart below, the asserted claims of the '560 patent are invalid under 35 U.S.C. § 102(a) and/or (b)[1] as anticipated by Narinder Singh, et al, "A Distributed and Autonomous Knowledge Sharing Approach to Software Interoperation" (March 22, 1995) ("Singh"), and/or are invalid under 35 U.S.C. § 103 as obvious in view Singh, or in combination with the reference(s) specifically identified in the following claim chart or one or more other references identified in Defendants' Preliminary Invalidity Contentions ("Defendants' Invalidity Contentions").

The additional KIF/KQML documents relied upon herein include, but are not limited to:

- Yannis Labrou, et al., "Semantics for an Agent Communication Language" (1997)[2] ("Labrou"). Labrou provides further descriptions and examples of the types of syntax and expressions possible with the KIF/KQML architecture that is described in Singh.

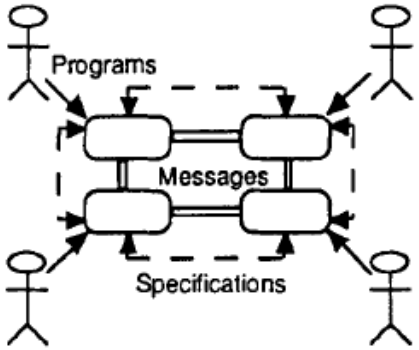
To the extent a finder of fact determines that the references cited herein do not teach certain limitations in the asserted claims, such limitations would have been inherent and/or obvious. These claims are also invalid as obvious in view of Singh alone or in combination with other prior art references, including, but not limited, to the prior art identified in the Cover Pleading of Defendants' Invalidity Contentions, the prior art described in the claim charts attached in Appendix C, and/or the prior art identified in Appendix D.

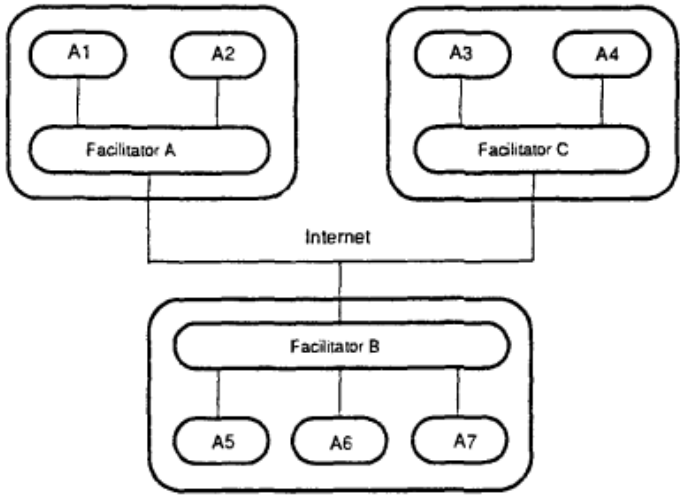
Defendants' Invalidity Contentions are based, in part, upon Defendants' present understanding of the asserted claims and IPA's apparent interpretations of the asserted claims in its July 10, 2019 Preliminary Infringement Contentions ("Infringement Contentions") and Defendants' investigation to date. Defendants are not adopting IPA's constructions or apparent constructions, nor are Defendants admitting to the accuracy of any particular contention or construction. The citations provided in the charts below are exemplary rather than exhaustive and Defendants reserve the right to rely upon additional references uncovered through further searching, other portions of the cited references and/or other portions of references cited within these Invalidity Contentions. Defendants further incorporate by reference the reservation of rights identified in the cover pleading to these Invalidity Contentions as though fully set

[1] Singh was publicly disclosed March 22, 1995 and published December 1995, more than one year before the January 5, 1999 earliest possible filing date of the '560 patent.
[2] Published in Munindar P. Singh, et al., Intelligent Agents IV Agent Theories, Architectures, and Languages, ATAL '97, vol 1365.

forth herein.

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(p)	A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of	Singh discloses a computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of claim 1. *See* Claim Chart for U.S. Patent No. 6,851,115 in view of Singh, Ex. A-8 (“’115 chart”), claim 1.
1(a)	a plurality of service-providing electronic agents;	Singh discloses a plurality of service-providing electronic agents. *See* ’115 chart, claim 1.
1(b)	a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:	Singh discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,* Singh at 339 (disclosing that an agent’s functionality can be distributed across multiple processes) (emphasis added). > [An] approach [supports] software interoperation based on specification sharing. Software components, called **agents**, provide machine processable descriptions of their capabilities and needs. Agents can be realized in different programming languages, and they **can run in different processes on different machines**. In addition, agents can be dynamic - at run time agents can join the system or leave. The system uses the declarative agent specifications to automatically coordinate their interoperation. *See also*, Singh at 341, FIG. 1 (disclosing the agents are in bi-directional communications with facilitators) (emphasis added). > Our efforts . . . rely on a highly expressive communication language called ACL (for Agent Communication Language). Programs (called agents) use ACL to supply machine-processable documentation to

	’560 Patent Claim Language	Invalidity in View of Prior Art
		system programs (called facilitators), which then coordinate their activities. The agent-based approach to **interoperability** is based on the notion of shared abstraction . . . Individual programmers can write their programs without knowledge of the specific vocabulary or interfaces of other software components. Computer users can avail themselves of the **services of different programs** by asking their systems to **coordinate their interaction**. The view of **automated coordination** is illustrated in the right half of Fig. 1 (reproduced below).  *See also* Singh at 363-64 (disclosing that the communication between agents and facilitators are bi-directional). . . . Whenever a new piece of information is added to the product agent's knowledge base it notifies the facilitator of it . . . The facilitator performs inference to see if any agent is interested . . . The facilitator cannot answer these questions locally, and it forwards the queries to the product-agent who can answer them . . . The facilitator passes on the two queries of the scheduler to the sm-datebook agent and the fm-synchronize agent.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		See also, Singh at 353. Figure 5 shows a picture of a Federation Architecture in the simple case where there are three machines, with one facilitator per machine, one machine with three agents, and the remaining with two agents each. Agents are restricted to communicate directly with facilitators. There can be an arbitrary number of facilitators, on one or more machines, and the network of facilitators can be connected arbitrarily.  Fig. 5. Federated system.
1(c)	an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment; and	Singh discloses an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment. *See* ’115 chart, claim 1(a).

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(d)	a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves:	Singh discloses a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves. *See* ’115 chart, claims 1(e), (h).
1(e)	using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal; and	Singh discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* ’115 chart, claim 1(g)-(i).
1(f)	wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes:	Singh discloses wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes. *See* ’115 chart, claims 1(a)-(c).
1(g)	a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events.	Singh discloses a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events. *See* ’115 chart, claims 1(b)-(c).

EXHIBIT D

Exhibit C-10

Invalidity of U.S. Patent No. 7,069,560 ("'560 Patent")
by Genesereth '94

As shown in the claim chart below, the asserted claims of the '560 patent are invalid under 35 U.S.C. § 102(a) and/or (b)[1] as anticipated by Michael Genesereth, et al., "Software Agents" (1994)[2] ("Genesereth '94"), and/or are invalid under 35 U.S.C. § 103 as obvious in view of Genesereth '94, or in combination with the reference(s) specifically identified in the following claim chart or one or more other references identified in Defendants' Preliminary Invalidity Contentions ("Defendants' Invalidity Contentions").

To the extent a finder of fact determines that the references cited herein do not teach certain limitations in the asserted claims, such limitations would have been inherent and/or obvious. These claims are also invalid as obvious in view of Genesereth '94 alone or in combination with other prior art references, including, but not limited, to the prior art identified in the Cover Pleading of Defendants' Invalidity Contentions, the prior art described in the claim charts attached in Appendix C, and/or the prior art identified in Appendix D.

Defendants' Invalidity Contentions are based, in part, upon Defendants' present understanding of the asserted claims and IPA's apparent interpretations of the asserted claims in its July 10, 2019 Preliminary Infringement Contentions ("Infringement Contentions") and Defendants' investigation to date. Defendants are not adopting IPA's constructions or apparent constructions, nor are Defendants admitting to the accuracy of any particular contention or construction. The citations provided in the charts below are exemplary rather than exhaustive and Defendants reserve the right to rely upon additional references uncovered through further searching, other portions of the cited references and/or other portions of references cited within these Invalidity Contentions. Defendants further incorporate by reference the reservation of rights identified in the cover pleading to these Invalidity Contentions as though fully set forth herein.

[1] Genesereth '94 was published in 1994, more than one year before the January 5, 1999 earliest possible filing date of the '560 patent.
[2] Published in Communications of the ACM, Volume 37, Number 7 (1994).

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(p)	A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of	To the extent that the preamble is held to be limiting, Genesereth ’94 discloses a computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of claim 1. *See* Claim Chart for U.S. Patent No. 6,851,115 in view of Genesereth ’94, Ex. A-10 (“’115 chart”), claim 1.
1(a)	a plurality of service-providing electronic agents;	Genesereth ’94 discloses a plurality of service-providing electronic agents. *See* ’115 chart, claim 1.
1(b)	a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:	Genesereth ’94 discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,* Genesereth ’94 at 51-52. “Once we have a language and the ability to build agents, there remains the question of how these agents should be organized to enhance collaboration. Two very different approaches have been explored: direct communication, in which agents handle their own coordination and assisted coordination, in which agents rely on special system programs to achieve coordination. The advantage of direct communication is that it does not rely on the existence, capabilities, or biases of any other programs.” *See also,* Genesereth ’94 at 53. “To deal with notational incompatibilities, facilitators can translate messages from one vocabulary to another using definitions supplied by agents or retrieved from the ACL dictionary. In so doing, they can decompose messages into submessages and send them to different agents. When necessary, they can combine multiple messages. In some cases this assistance can be rendered

	’560 Patent Claim Language	Invalidity in View of Prior Art
		interpretively with messages going through the facilitators. In other cases, it can be done in one-shot fashion with the facilitators setting up specialized links among individual agents and then stepping out of the picture.” *See also,* Genesereth ’94 at 52. “A popular alternative to direct communication that eliminates both of these disadvantages is to organize agents into what is often called a federated system. Figure 2 illustrates the structure of such a system in the simple case in which there are just three machines, one with three agents and two with two agents apiece. As suggested by the diagram, agents do not communicate directly with one another. Instead, they communicate only with system programs called facilitators, and facilitators communicate with one another. (The concept of a facilitator [6] derives from and generalizes the concept of a mediator [16].) In a federated system, agents use ACL (in practice, a restricted subset of ACL) to document their needs and abilities for their local facilitators. In addition to this metalevel information, they also send application-level information and requests to their facilitators and accept application-level information and requests in return. Facilitators use the documentation provided by these agents to transform these application-level messages and route them to the appropriate places.” *See also,* Genesereth ’94 at 52-53. “The concepts of system services in support of software interoperation are not new here. For example, directory assistance programs facilitate software interoperation by providing a way for programs to discover which programs can handle which requests and which programs are interested in which pieces of information. Distributed object managers such as CORBA, OLE, DSOM provide location transparency for object-oriented systems, routing messages to objects without requiring senders to know the locations of those objects. Automatic brokers such as the Publish and Subscribe capabilities on the Macin-tosh, DDE, BMS, and Tooltalk, combine these capabilities—they not only compute the appropriate programs to receive messages but forward those

	’560 Patent Claim Language	Invalidity in View of Prior Art
		messages, handle any problems that arise, and, where appropriate, return the answers to the original senders.”
1(c)	an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment; and	Genesereth ’94 discloses an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment. *See* ’115 chart, claim 1(a).
1(d)	a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves:	Genesereth ’94 discloses a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves. *See* ’115 chart, claims 1(e), (h).
1(e)	using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal; and	Genesereth ’94 discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* ’115 chart, claim 1(g)-(i). *See, e.g.*, Genesereth ’94 at 53 (regarding “using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal”). “To provide these capabilities, current implementations of facilitators take advantage of automated reasoning technology developed in the artificial intelligence (AI) and database communities. Powerful search control techniques are used to enhance normal message-processing performance, and automatic generation of message-routing programs and pair-wise translators is used for cases requiring greater efficiency.”

	’560 Patent Claim Language	Invalidity in View of Prior Art
		See also, Genesereth ’94 at 53. “The primary difference between these approaches to software interoperation and agent-based software engineering lies in the sophistication of the processing done by facilitators. Using ACL, agents can express their needs and capabilities more accurately than in pattern-based metalanguages, and facilitators can use this added information to be more discriminating in routing messages. To deal with notational incompatibilities, facilitators can translate messages from one vocabulary to another using definitions supplied by agents or retrieved from the ACL dictionary. In so doing, they can decompose messages into submessages and send them to different agents. When necessary, they can combine multiple messages. In some cases this assistance can be rendered interpretively with messages going through the facilitators. In other cases, it can be done in one-shot fashion with the facilitators setting up specialized links among individual agents and then stepping out of the picture.”
1(f)	wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes:	Genesereth ’94 discloses wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes. *See* ’115 chart, claims 1(a)-(c).
1(g)	a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events.	Genesereth ’94 discloses a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events. *See* ’115 chart, claims 1(b)-(c).
20	A computer architecture as recited in claim 1 wherein the distributed	Genesereth ’94 discloses a computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first

EXHIBIT E

Exhibit C-14

Invalidity of U.S. Patent No. 7,069,560 ("'560 Patent")
by Finin I

As shown in the claim chart below, the asserted claims of the '560 patent are invalid under 35 U.S.C. § 102(a) and/or (b)[1] as anticipated by Tim Finin, "Software Agents Knowledge Sharing KQML, KIF and Ontologies" (October 16, 1997), *available at* https://www.csee.umbc.edu/~finin/sisce97/ ("Finin I"), and/or are invalid under 35 U.S.C. § 103 as obvious in view of Finin I, or in combination with the reference(s) specifically identified in the following claim chart or one or more other references identified in Defendants' Preliminary Invalidity Contentions ("Defendants' Invalidity Contentions").

The additional KIF/KQML documents relied upon herein include, but are not limited to:

- Yannis Labrou, et al., "Semantics for an Agent Communication Language" (1997)[2] ("Labrou"). Labrou provides further description and examples of the type of syntax and expressions possible with the KIF/KQML architecture described in Finin I.

To the extent a finder of fact determines that the references cited herein do not teach certain limitations in the asserted claims, such limitations would have been inherent and/or obvious. These claims are also invalid as obvious in view of Finin I alone or in combination with other prior art references, including, but not limited, to the prior art identified in the Cover Pleading of Defendants' Invalidity Contentions, the prior art described in the claim charts attached in Appendix C, and/or the prior art identified in Appendix D.

Defendants' Invalidity Contentions are based, in part, upon Defendants' present understanding of the asserted claims and IPA's apparent interpretations of the asserted claims in its July 10, 2019 Preliminary Infringement Contentions ("Infringement Contentions") and Defendants' investigation to date. Defendants are not adopting IPA's constructions or apparent constructions, nor are Defendants admitting to the accuracy of any particular contention or construction. The citations provided in the charts below are exemplary rather than exhaustive and Defendants reserve the right to rely upon additional references uncovered through further searching, other portions of the cited references and/or other portions of references cited within these Invalidity Contentions. Defendants further incorporate by reference the reservation of rights identified in the cover pleading to these Invalidity Contentions as though fully set forth herein.

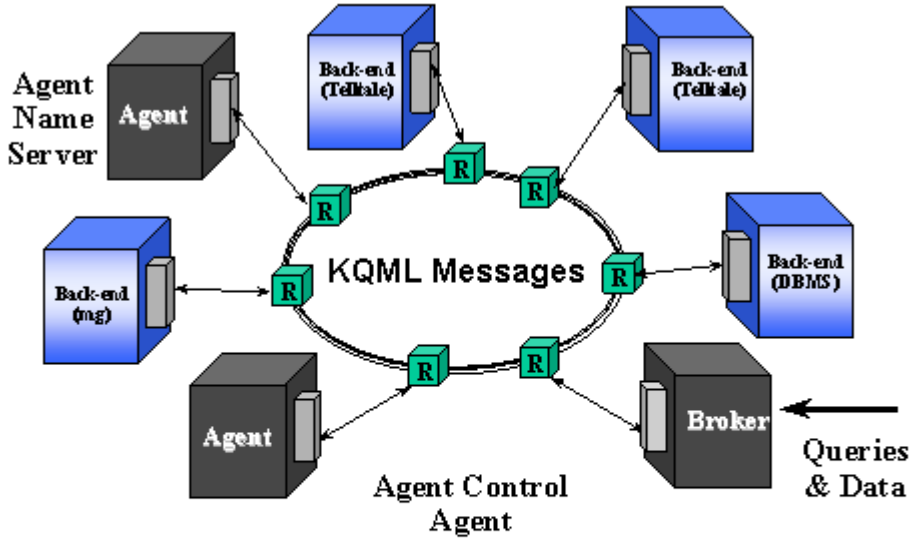
---

[1] Finin I was publicly available at least as early as October 16, 1997, more than one year before the January 5, 1999 earliest possible filing date of the '560 patent.

[2] Published in Munindar P. Singh, et al., Intelligent Agents IV Agent Theories, Architectures, and Languages, ATAL '97, vol 1365.

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(p)	A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of	To the extent that the preamble is held to be limiting, Finin I discloses a computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of claim 1. *See* Claim Chart for U.S. Patent No. 6,851,115 in view of Finin I, Ex. A-14 (“’115 chart”), claim 1.
1(a)	a plurality of service-providing electronic agents;	Finin I discloses a plurality of service-providing electronic agents. *See* ’115 chart, claim 1.
1(b)	a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:	Finin I discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,* Finin I at 86.

	'560 Patent Claim Language	Invalidity in View of Prior Art
		Some key ideas • Software agents offer a new paradigm for very large scale distributed heterogeneous applications focusing on the interactions of autonomous, cooperating processes which can adapt to humans and other agents. • Intelligence is always a desirable characteristic but is not strictly required by the paradigm. • There is a wealth of prior theory from several disciplines but you needn't to be an expert to to apply or use it. Slide 86 of 87 *See also*, Finin I at 74.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 Slide 74 of 87 Finin I discloses this limitation as identified above. This limitation is also obvious in view of Finin I combined with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X. As Ex. C-X shows, each of these references also discloses this limitation. A person having ordinary skill in the art would have been motivated to combine Finin I with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X rendering this limitation obvious based on one or more of the motivations to combine identified in Section II.A.3.e of the cover pleading to Defendants’

	’560 Patent Claim Language	Invalidity in View of Prior Art
		Preliminary Invalidity Contentions and because each of these references relate to the same field of software agent technology and distributed computing environments.
1(c)	an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment; and	Finin I discloses an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment. *See* ’115 chart, claim 1(a).
1(d)	a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves:	Finin I discloses a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves. *See* ’115 chart, claims 1(e), (h).
1(e)	using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal; and	Finin I discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* ’115 chart, claim 1(g)-(i).
1(f)	wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes:	Finin I discloses wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes. *See* ’115 chart, claims 1(a)-(c).

	'560 Patent Claim Language	Invalidity in View of Prior Art
1(g)	a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events.	Finin I discloses a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events. *See* '115 chart, claims 1(b)-(c).
20	A computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process.	Finin I discloses this limitation as identified above. This limitation is also obvious in view of Finin I combined with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X. As Ex. C-X shows, each of these references also discloses this limitation. A person having ordinary skill in the art would have been motivated to combine Finin I with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X rendering this limitation obvious based on one or more of the motivations to combine identified in Section II.A.3.e of the cover pleading to Defendants' Preliminary Invalidity Contentions and because each of these references relate to the same field of software agent technology and distributed computing environments.
21	A computer architecture as recited in claim 20 wherein the planning component is one of a plurality of synchronized planning components each executing with separate computer processes, whereby the computer architecture provides a more robust operating environment due to redundancy of the planning component	Upon information and belief, Finin I discloses computer architecture as recited in claim 20 wherein the planning component is one of a plurality of synchronized planning components each executing with separate computer processes, whereby the computer architecture provides a more robust operating environment due to redundancy of the planning component functionality of the distributed facilitator agent. Finin I discloses this limitation as identified above. This limitation is also obvious in view of Finin I combined with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X. As Ex. C-X shows, each of these references also discloses this limitation. A person having ordinary skill in the art would have been motivated to

EXHIBIT F

Exhibit C-20

Invalidity of U.S. Patent No. 7,069,560 ("'560 Patent")
by InfoSleuth

As shown in the claim chart below, the asserted claims of the '560 patent are invalid under 35 U.S.C. § 102(a)[1] as anticipated by "Facilitating Open Communication in Agent Systems: The InfoSleuth Infrastructure," by Nodine et al. in Intelligent Agents IV: Agent Theories, Architectures, and Languages, Proceedings 4th International Workshop, ATAL'97 ("Nodine"), and/or are invalid under 35 U.S.C. § 103 as obvious in view of Nodine, or in combination with the reference(s) specifically identified in the following claim chart or one or more other references identified in Defendants' Preliminary Invalidity Contentions ("Defendants' Invalidity Contentions").

The Secondary References relied upon herein include, but are not limited to:

- "Experience with the InfoSleuth Agent Architecture" by Nodine et al. in the Proceedings of the Fifteenth National Conference on AI, September 27, 1998 ("Nodine 2") published on September 27, 1998
- "InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments" by Bayardo et al. in the Proceedings ACM SIGMOD International Conference on Management of Data, Vol. 26, Issue 2 ("Bayardo") published June 1997
- "Expressing Composite Events in InfoSleuth" by Urban et al. ("Urban") published December 1998

Nodine, Nodine 2, Bayardo, and Urban references collectively, "InfoSleuth," which are collectively prior art under at least § 102(a), § 102(b), and/or § 102(g).

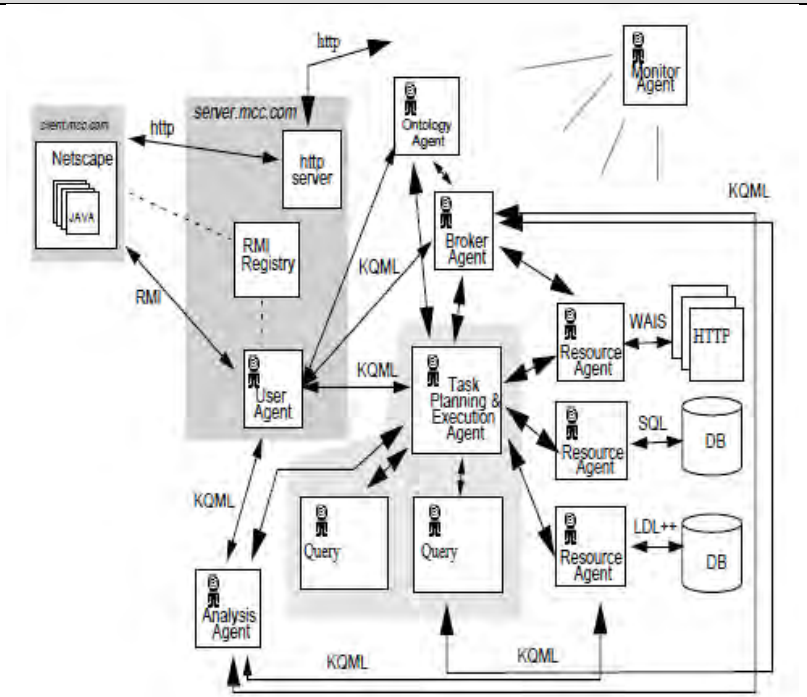
To the extent a finder of fact determines that the references cited herein do not teach certain limitations in the asserted claims, such limitations would have been inherent and/or obvious. These claims are also invalid as obvious in view of Nodine, Nodine 2, Bayardo, and/or Urban (and/or InfoSleuth) alone or in combination with other prior art references, including, but not limited, to the prior art identified in the Cover Pleading of Defendants' Invalidity Contentions, the prior art described in the claim charts attached in Appendix C, and/or the prior art identified in Appendix D.

Defendants' Invalidity Contentions are based, in part, upon Defendants' present understanding of the asserted claims and IPA's apparent interpretations of the asserted claims in its July 10, 2019 Preliminary Infringement Contentions ("Infringement Contentions") and Defendants' investigation to date. Defendants are not adopting IPA's constructions or apparent constructions, nor are Defendants

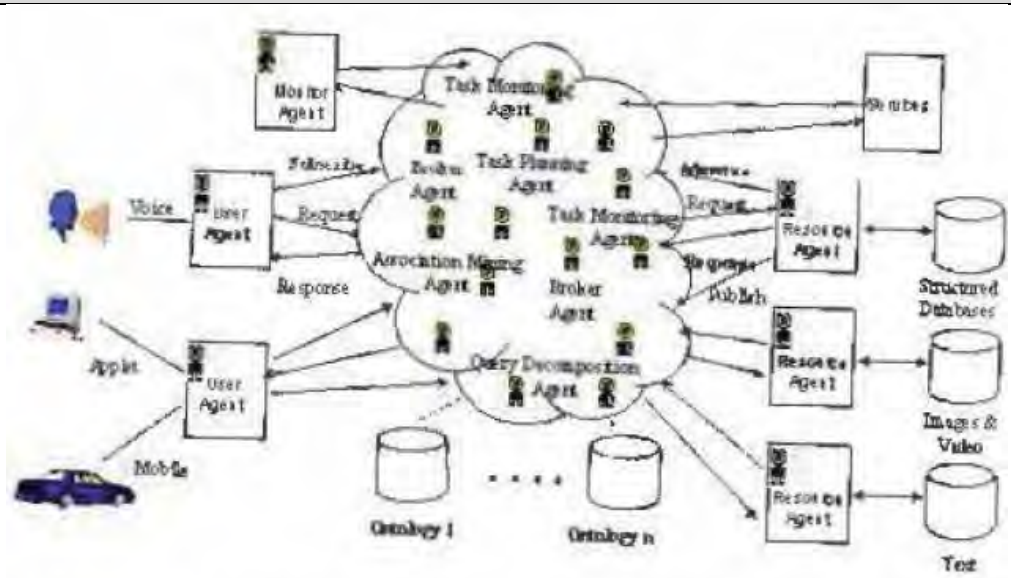
[1] InfoSleuth was published at least in 1998 before the January 5, 1999 earliest possible filing date of the '560 patent.

admitting to the accuracy of any particular contention or construction. The citations provided in the charts below are exemplary rather than exhaustive and Defendants reserve the right to rely upon additional references uncovered through further searching, other portions of the cited references and/or other portions of references cited within these Invalidity Contentions. Defendants further incorporate by reference the reservation of rights identified in the cover pleading to these Invalidity Contentions as though fully set forth herein.

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(p)	A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of	To the extent the preamble is found to be limiting, InfoSleuth discloses a computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of claim 1. *See* Claim Chart for U.S. Patent No. 6,851,115 in view of InfoSleuth, Ex. A-20 (“’115 chart”), claim 1.
1(a)	a plurality of service-providing electronic agents;	InfoSleuth discloses a plurality of service-providing electronic agents. *See* ’115 chart, claim 1.
1(b)	a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:	InfoSleuth discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,*’115 chart, claims 1(a), (e), (h), 11; *see also, e.g.*,

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 Figure 1: The InfoSleuth architecture Bayardo at Figure. We intend to enhance the brokering capabilities, splitting the broker agent into a family of cooperating, specialized brokers. We will factor out the syntactic brokering capabilities into a separate type of broker agent, possibly implementing it as an ORB interface using CORBA [30]. Semantic brokering will be available at different levels—for example, local to the site, local to the enterprise, and between enterprises. Semantic brokering may include additional information on contents, and additional semantic information such as quality and cost of information.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		Bayardo at 205. > We are in the process of splitting the current execution agent into two separate agents, a query decomposition agent and a task execution agent. The task execution agent will develop execution plans based on user requirements using generative planning and plan retrieval utilizing case-based reasoning techniques [17, 31]. The task execution agent may interleave planing with information-gathering subtasks [2, 34, 23] and repair plans when unexpected situations are encountered [10, 26]. Plans will be specified as (transactional) work flows that can be executed by InfoSleuth. It will supervise the execution of the resulting work flows, including managing the transactions it generates. The query decomposition agent will be called by the task execution agent when it has a query over multiple resource agents. It will optimize and decompose queries over multiple resource agents, reassemble the results, and return them to the task execution agent. Bayardo at 205. > After a query is formulated in terms of the selected common ontology, it is sent to the task execution agent that best meets the user's needs with respect to the current query context. Bayardo at 197.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 Figure 1: InfoSleuth: Dynamic and Broker-based Agent Architecture. Nodine 2 at Figure 1. A person or ordinary skill in the art would have also recognized that a distributed facilitator agent functionally distributed across at least two computer processes is obvious. Such a skilled person would have recognized that the above configuration of the facilitator agent would have promoted efficient computation via decomposition/parallelization of tasks (consistent with service/task fulfillment approach predicated on decomposition and delegation of tasks), ease of control/administration of the facilitator, and/or redundancy/fault tolerance. InfoSleuth discloses this limitation as demonstrated above. A person of ordinary skill in the art would have also been motivated to combine the InfoSleuth with another reference, including, for example, the references discussed in Ex. C-X that disclose a

	’560 Patent Claim Language	Invalidity in View of Prior Art
		distributed facilitator agent functionally distributed across at least two computer processes, at least because doing so would have required nothing more than combining known elements within the creativity of such a person of ordinary skill in the art and the combination would involve assembling the known elements in a predictable way to produce predictable results. Both InfoSleuth and the references discussed in Ex. C-X are directed to a common field of endeavor and include/disclose/discuss intelligent multi-agent systems and/or distributed computer systems and architectures. A person having ordinary skill in the art would be motivated to combine InfoSleuth with the references discussed in Ex. C-X at least because they discuss more efficient means for inter-agent communication and/or the above configuration of the facilitator agent would have promoted efficient computation via decomposition/parallelization of tasks (consistent with service/task fulfillment approach predicated on decomposition and delegation of tasks), ease of control/administration of the facilitator, and/or redundancy/fault tolerance. InfoSleuth discloses this limitation as identified above. This limitation is also obvious in view of InfoSleuth combined with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X. As Ex. C-X shows, each of these references also discloses this limitation. A person having ordinary skill in the art would have been motivated to combine InfoSleuth with one or more of the references identified in the corresponding limitation of Ex. C-X rendering this limitation obvious based on one or more of the motivations to combine identified in Section II.A.3.E of the cover pleading to Defendants’ Preliminary Invalidity Contentions and because each of these references relate to the same field of software agent technology and distributed computing environments.
1(c)	an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment; and	InfoSleuth discloses an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment. *See* ’115 chart, claim 1(a).

	'560 Patent Claim Language	Invalidity in View of Prior Art
1(d)	a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves:	InfoSleuth discloses a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves. *See* '115 chart, claims 1(e), (h).
1(e)	using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal; and	InfoSleuth discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* '115 chart, claim 1(g)-(i). > An agent must be able to forward a request to another agent for handling and reply. Similarly, an agent must be able to delegate by initiating a request but specifying that the recipient send its reply elsewhere. . . . Such a situation often arises in complex tasks, where an agent handles part of a request itself, but delegates some aspect of it to another. For example, InfoSleuth supports knowledge mining activities in which a task execution agent receives a knowledge mining task from a user agent. As part of accomplishing that task, the task execution agent sends a series of messages to a knowledge mining agent. The knowledge mining agent's replies, as well as additional replies by the task execution agent, are all returned to the user agent in response to its original message. Nodine 2 at 68.

	’560 Patent Claim Language	Invalidity in View of Prior Art

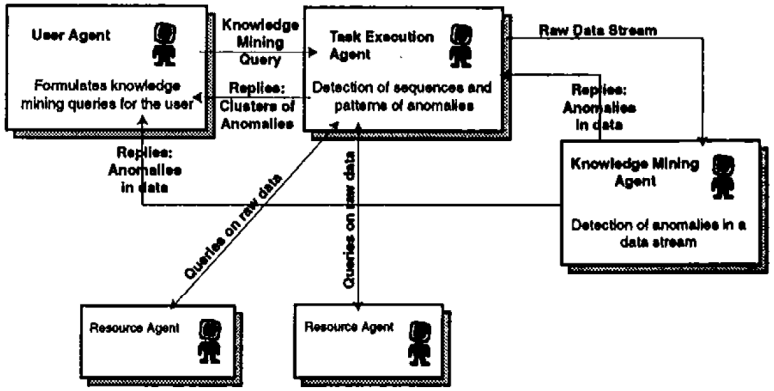


Figure 4: Delegation in support of a query, and replies to the query returned to a user agent from both a task execution agent and a knowledge mining agent.

Nodine 2 at Figure 4.

> Task Execution Agent: coordinates the execution of high-level information-gathering subtasks (scenarios) necessary to fulfill the queries. It uses information supplied by the Broker Agent to identify the resources that have the requested information, routes requests to the appropriate Resource Agents, and reassembles the results.

Bayardo at 197; Bayardo at 198 (“The Task Execution Agent coordinates the execution of high-level information gathering tasks.”); Bayardo at 198 (“Task Plan Execution Using Domain-independent Rules. After an agent’s knowledge base has been populated with operator descriptions and declarative task plans, it uses its domain-independent task execution knowledge to carry out the plans.”)

	’560 Patent Claim Language	Invalidity in View of Prior Art
		The Task Execution Agent coordinates the execution of high-level information gathering tasks. We use the term “high-level” to suggest work flow-like or data mining and analysis activities. Such high-level tasks can potentially include global query decomposition and post-processing as sub-tasks carried out by decomposition sub-agents, where the global query is couched in terms of a common ontology; and subqueries must be generated based on the schemas and capabilities of the various resources known to the system, and then the results joined. Bayardo at 198. Each time a query from the user agent is received, a new instantiation of the appropriate plan from the plan library is initialized by the rule-based system. . . . The sequences of interactions with other agents are determined by the task plans the agent executes . . . . Bayardo at 198; *Id.* (“The approach we have taken for the Task Execution Agent is based on the use of declarative task plans. . . .”) Example: General Query Task Plan. Executing a general query task plan causes the Task Execution Agent to carry out the following step. * Advertise to the Broker, using a tell performative, and wait to receive a reply (done at agent initialization). * Wait to receive queries from User Agents. These will typically be encoded as KQML directives, such as ask-all, standby, or subscribe). The query as well as the domain context determines the task plan that is instantiated to process the query. * Parse the query, and decompose it if appropriate. Parsing involves getting an ontological model from the Ontology Agent; once this model is obtained, it is cached for future use.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		* Construct KIF queries based on the SQL queries' contents, and query the Broker using the KIF queries and the ask-all performative to find relevant resources. * Query the relevant resource agents specified by the broker. * Compose the results. * Incrementally return the results to the user agent using a streaming protocol. Using this protocol, the user agent successively requests additional result tuples. Bayardo at 198. The Broker Agent determines the set of relevant resources that can perform the requested service. As agents come on line, they advertise their services to the broker via KQML. The Broker Agent responds to an agent's request for service with information about the other agents that have previously advertised relevant service. Details of the Broker protocols describing the exchanged information are given in section 5.2. In effect, the Broker Agent is a cache of metadata that optimizes access in the agent network. Any individual agent could perform exactly the same queries on an as-needed basis. In addition, the existence of the Broker Agent both reduces the individual agent's need for knowledge about the structure of the network and decreases the amount of network traffic required to accomplish an agent's task. Minimally, an agent must advertise to the Broker its location, name, and the language it speaks. Additionally, agents may advertise meta-information and domain constraints based on which it makes sense to query a given agent. The purpose of domain advertising is to allow the Broker to reason about queries and to rule out those queries which are known to return null results. For example, if a Resource Agent advertises that it knows about only those medical procedures relating to heart surgery, it is inappropriate to query it regarding liver resection, and the Broker would not recommend it to an agent seeking liver resection data.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		Bayardo at 199. An agent’s advertisement of content is in terms of portions of domain-specific common ontologies, so that an agent may constrain its capability advertisement to apply to only a select set of concepts, relationships, or instances from a particular application domain. . . . The domain ontology defines a set of ‘domain events and activities’ that drive decision making in the application. Nodine 2 at 64. Connect any application-specific logic to the InfoSleuth agent shell and advertise it in the network as an agent ready to provide information services on fragments of the domain ontology. Nodine 2 at 69. In all cases, we are experiencing very rapid “time to deployment” of complex and domain-specific information gathering and analysis applications. Nodine 2 at 70. *See also* claim 1(a); Nodine 2 at 64, 71 (“[t]he InfoSleuth system dynamically constructs information gathering agent communities, based on brokering and planning principles, to satisfy given tasks as best as possible” and that “[t]he InfoSleuth architecture allows agents to exploit rules for conversation when they have that ability. . . .”); Nodine 2 at 71 (“Task Planning”).
1(f)	wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes:	InfoSleuth discloses wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes. *See* ’115 chart, claims 1(a)-(c).
1(g)	a layer of conversational protocol defined by event types and parameter lists associated with one	InfoSleuth discloses a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events.

	’560 Patent Claim Language	Invalidity in View of Prior Art
	or more of the events, wherein the parameter lists further refine the one or more events.	*See* ’115 chart, claims 1(b)-(c).
20	A computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process.	InfoSleuth discloses a computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process. *See, e.g.,* Claim Chart for U.S. Patent No. 6,851,115 (“’115 chart”), claims 1(a), (e), (h); *see also, e.g.*, > We intend to enhance the brokering capabilities, splitting the broker agent into a family of cooperating, specialized brokers. We will factor out the syntactic brokering capabilities into a separate type of broker agent, possibly implementing it as an ORB interface using CORBA [30]. Semantic brokering will be available at different levels—for example, local to the site, local to the enterprise, and between enterprises. Semantic brokering may include additional information on contents, and additional semantic information such as quality and cost of information. Bayardo at 205. > We are in the process of splitting the current execution agent into two separate agents, a query decomposition agent and a task execution agent. The task execution agent will develop execution plans based on user requirements using generative planning and plan retrieval utilizing case-based reasoning techniques [17, 31]. The task execution agent may interleave planing with information-gathering subtasks [2, 34, 23] and repair plans when unexpected situations are encountered [10, 26]. Plans will be specified as (transactional) work flows that can be executed by InfoSleuth. It will supervise the execution of the resulting work flows, including managing the transactions it generates. The query decomposition agent will be called by the task execution agent when it has a

EXHIBIT G

Exhibit C-23

Invalidity of U.S. Patent No. 7,069,560 ("'560 Patent")
by Finin II

As shown in the claim chart below, the asserted claims of the '115 patent are invalid under 35 U.S.C. §§ 102(a) and/or (b)[1] as anticipated by "KQML as an Agent Communication Language" by Tim Finin, et al. ("Finin II"), and/or are invalid under 35 U.S.C. § 103 as obvious in view of Finin II, or in combination with the reference(s) specifically identified in the following claim chart or one or more other references identified in Defendants' Preliminary Invalidity Contentions ("Invalidity Contentions").

To the extent a finder of fact determines that the references cited herein do not teach certain limitations in the asserted claims, such limitations would have been inherent and/or obvious. These claims are also invalid as obvious in view of Finin II alone or in combination with other prior art references, including, but not limited, to the prior art identified in the Cover Pleading of Defendants' Invalidity Contentions, the prior art described in the claim charts attached in Appendix C, and/or the prior art identified in Appendix D.

Defendants' Invalidity Contentions are based, in part, upon Defendants' present understanding of the asserted claims and IPA's apparent interpretations of the asserted claims in its July 10, 2019 Preliminary Infringement Contentions ("Infringement Contentions") and Defendants' investigation to date. Defendants are not adopting IPA's constructions or apparent constructions, nor are Defendants admitting to the accuracy of any particular contention or construction. The citations provided in the charts below are exemplary rather than exhaustive and Defendants reserve the right to rely upon additional references uncovered through further searching, other portions of the cited references and/or other portions of references cited within these Invalidity Contentions. Defendants further incorporate by reference the reservation of rights identified in the cover pleading to these Invalidity Contentions as though fully set forth herein.

[1] Finin II was published in Jeffrey M. Bradshaw, "Software Agents," *American Association for Artificial Intelligence* (1997), before the January 5, 1999 earliest possible priority date of the '560 patent.

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(p)	A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of	To the extent the preamble is found to be limiting, Finin II discloses a computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of claim 1. *See* Claim Chart for U.S. Patent No. 6,851,115 in view of Finin II, Ex. A-23 (“’115 chart”), claim 1.
1(a)	a plurality of service-providing electronic agents;	Finin II discloses a plurality of service-providing electronic agents. *See* ’115 chart, claim 1.
1(b)	a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:	Finin II discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,* Finin II at 13 (emphasis added). > A facilitator is an agent that performs various useful communication services, e.g. maintaining a registry of service names, forwarding messages to named services, routing messages based on content, providing “**matchmaking**" between information providers and clients, and providing mediation and translation services. *See also,* Finin II at 17.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		The design and engineering of complex computer systems, whether exclusively hardware or software systems or both, today involves multiple design and engineering disciplines. Many such systems are developed in fast cycle or concurrent processes which involve the immediate and continual consideration of end–product constraints, e.g., marketability, manufacturing planning, etc. Further, the design, engineering and manufacturing components are also likely to be distributed across organizational and company boundaries. KQML has proved highly effective in the integration of diverse tools and systems enabling new tool interactions and supporting a high–level communication infrastructure reducing integration cost as well as flexible communication across multiple networking systems. The use of KQML in these demonstrations has allowed the integrators to focus on what the integration of design and engineering tools can accomplish and appropriately deemphasized how the tools communicate [14, 22, 8, 9]. Finin II discloses this limitation as identified above. This limitation is also obvious in view of Finin II combined with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X. As Ex. C-X shows, each of these references also discloses this limitation. A person having ordinary skill in the art would have been motivated to combine Finin II with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X rendering this limitation obvious based on one or more of the motivations to combine identified in Section II.A.3.e of the cover pleading to Defendants’ Preliminary Invalidity Contentions and because each of these references relate to the same field of software agent technology and distributed computing environments.
1(c)	an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment; and	Finin II discloses an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment. *See* ’115 chart, claim 1(a).
1(d)	a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a	Finin II discloses a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves.

	’560 Patent Claim Language	Invalidity in View of Prior Art
	goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves:	*See* ’115 chart, claims 1(e), (h).
1(e)	using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal; and	Finin II discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* ’115 chart, claim 1(g)-(i). Finin II discloses this limitation as identified above. This limitation is also obvious in view of Finin II combined with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X. As Ex. C-X shows, each of these references also discloses this limitation. A person having ordinary skill in the art would have been motivated to combine Finin II with one or more of the references identified in the corresponding limitation of Ex. C-X rendering this limitation obvious based on one or more of the motivations to combine identified in Section II.A.3.e of the cover pleading to Defendants’ Preliminary Invalidity Contentions and because each of these references relate to the same field of software agent technology and distributed computing environments.
1(f)	wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes:	Finin II discloses wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes. *See* ’115 chart, claims 1(a)-(c).
1(g)	a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events.	Finin II discloses a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events. *See* ’115 chart, claims 1(b)-(c).

EXHIBIT H

Exhibit C-25

Invalidity of U.S. Patent No. 7,069,560 ("'560 Patent")
by Labrou Thesis

As shown in the claim chart below, the asserted claims of the '560 patent are invalid under 35 U.S.C. § 102(a) and/or (b)[1] as anticipated by Yannis Labrou, "Semantics for an Agent Communication Language" (August 1996) ("Labrou Thesis"), and/or are invalid under 35 U.S.C. § 103 as obvious in view of Labrou Thesis, or in combination with the reference(s) specifically identified in the following claim chart or one or more other references identified in Defendants' Preliminary Invalidity Contentions or subsequent Supplemental Invalidity Contentions (collectively "Defendants' Invalidity Contentions").

The additional KIF/KQML documents relied upon herein include, but are not limited to:

- "handler1.pl" by Yannis Labrou ("Handler1")
- "Software Agents Knowledge Sharing KQML, KIF and Ontologies" by Tim Finin (Finin I) and "KQML as an Agent Communication Language" by Tim Finin, et al. ("Finin II"). Finin I and Finin II provide further description and examples of the type of syntax and expressions possible with the KIF/KQML architecture described in Labrou Thesis.

To the extent a finder of fact determines that the references cited herein do not teach certain limitations in the asserted claims, such limitations would have been inherent and/or obvious. These claims are also invalid as obvious in view of Labrou Thesis alone or in combination with other prior art references, including, but not limited, to the prior art identified in the Cover Pleading of Defendants' Invalidity Contentions, the prior art described in the claim charts attached in Appendix C, and/or the prior art identified in Appendix D.

Defendants' Invalidity Contentions are based, in part, upon Defendants' present understanding of the asserted claims and IPA's apparent interpretations of the asserted claims in its July 10, 2019 Preliminary Infringement Contentions ("Infringement Contentions") and Defendants' investigation to date. Defendants are not adopting IPA's constructions or apparent constructions, nor are Defendants

[1] Labrou Thesis was publicly available at least as early as August 1996, more than one year before the January 5, 1999 earliest possible filing date of the '115 patent. Labrou Thesis is the dissertation of Yannis Labrou submitted to the Faculty of the Graduate School of the University of Maryland in partial fulfillment of the requirements for the degree of Doctor of Philosophy, reproduced from the microfilm master by UMI Company, copyright 1996. The catalog system of the University System of Maryland and Affiliated Institutions identifies Labrou Thesis as published in 1996 and available in the stacks of UMBC Library. *See* Labrou Thesis record, USMAI Library Catalog, available at https://catalog.umd.edu/docno=002378493; DEFS_IPA_PA00040915-20.

admitting to the accuracy of any particular contention or construction. The citations provided in the charts below are exemplary rather than exhaustive and Defendants reserve the right to rely upon additional references uncovered through further searching, other portions of the cited references and/or other portions of references cited within these Invalidity Contentions. Defendants further incorporate by reference the reservation of rights identified in the cover pleading to these Invalidity Contentions as though fully set forth herein.

	'560 Patent Claim Language	**Invalidity in View of Prior Art**
1(p)	A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of	To the extent the preamble is found to be limiting, Labrou Thesis discloses a computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of claim 1. *See* Claim Chart for U.S. Patent No. 6,851,115 in view of Labrou Thesis, Ex. A-25 ("'115 chart"), claim 1.
1(a)	a plurality of service-providing electronic agents;	Labrou Thesis discloses a plurality of service-providing electronic agents. *See* '115 chart, claim 1.
1(b)	a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:	Labrou Thesis discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,* Labrou Thesis at 26. > **Router**. The router handles all KQML messages going to and from its associated application. Each KQML-speaking software agent has its own router process but all routers are identical. Routers are content independent message routers that provide the agent with a single point of contact for the rest of the network. A router provides both client and server functions for the application and manages multiple simultaneous connections with other agents. *See also* Labrou Thesis at 79-80 (emphasis added).

	’560 Patent Claim Language	Invalidity in View of Prior Art
		• In each domain of KQML-speaking agents there is at least one agent with a special status called facilitator that can always handle the networking and facilitation performatives. Agents advertise to their facilitator, i.e., they send advertise messages to their facilitators, thus announcing the messages that they are committed to accepting and properly processing. Advertising to a facilitator is like advertising to the community (either of their own domain or of some other domain). Agents can still advertise on a one-to-one basis, if they so wish, and such advertisements do not commit them to processing messages from agents other than the : receiver of the advertise . Actually, such advertisements will never be shared with other agents, because of the “personal” nature of the advertisement, i.e., they are addressed to particular agents and only facilitators can supersede that: see Table 4.5. also. Agents can use their facilitator either - to have their queries properly dispatched to other agents, using recruit-one, recruit-all, broker-one or broker-all, or - to send a recommend-one or a recommend-all to get the relevant advertise messages and directly contact agent(s) that may process their queries. • Agents can access agents in other domains either through their facilitator, or directly. This implies that **a smart facilitator may be built in such a way that whenever it cannot find a useful, relevant advertise from an agent in its domain, it may query another facilitator, in some other domain**. Such an action initiates a sub-dialogue with another facilitator in order to serve the original query. Elaborate protocols of this kind are examples of conversations (interactions) that be built on top of the conversation policies presented in Chapter 6.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		• Facilitators may request the services of other facilitators in the same way that regular agents may request the services of their facilitator. Facilitators do not advertise, not even to other facilitators. The model we imply is one where regular agents advertise their services to their facilitators and thus facilitators become providers of query-processing information about the agents in their domain; such information can then be accessed by any agent (regular or facilitator), using the facilitation performatives. • We use the term facilitator to refer to all kinds of special services that may be provided by specialized agents, such as Agent Name Servers (ANS). proxy agents, or brokers ([34]). Labrou Thesis discloses this limitation as identified above. This limitation is also obvious in view of Labrou Thesis combined with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X. As Ex. C-X shows, each of these references also discloses this limitation. A person having ordinary skill in the art would have been motivated to combine Labrou Thesis with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X rendering this limitation obvious based on one or more of the motivations to combine identified in Section II.A.3.e of the cover pleading to Defendants’ Invalidity Contentions and because each of these references relate to the same field of software agent technology and distributed computing environments.
1(c)	an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment; and	Labrou Thesis discloses an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment. *See* ’115 chart, claim 1(a).

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(d)	a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves:	Labrou Thesis discloses a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves. *See* ’115 chart, claims 1(e), (h).
1(e)	using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal; and	Labrou Thesis discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* ’115 chart, claim 1(g)-(i). Labrou Thesis discloses this limitation as identified above. This limitation is also obvious in view of Labrou Thesis combined with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X. As Ex. C-X shows, each of these references also discloses this limitation. A person having ordinary skill in the art would have been motivated to combine Labrou Thesis with one or more of the references identified in the corresponding limitation of Ex. C-X rendering this limitation obvious based on one or more of the motivations to combine identified in Section II.A.3.e of the cover pleading to Defendants’ Invalidity Contentions and because each of these references relate to the same field of software agent technology and distributed computing environments.
1(f)	wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes:	Labrou Thesis discloses wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes. *See* ’115 chart, claims 1(a)-(c).

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(g)	a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events.	Labrou Thesis discloses a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events. *See* ’115 chart, claims 1(b)-(c).
20	A computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process.	Labrou Thesis discloses a computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process of claim 20. *See, e.g.,* Labrou Thesis at 5.

EXHIBIT I

Exhibit C-15

Invalidity of U.S. Patent No. 7,069,560 ("'560 Patent")
by *Kiss*

As shown in the claim chart below, the asserted claims of the '560 patent are invalid under at least 35 U.S.C. § 102(e)[1] as anticipated by U.S. Patent No. 6,484,155 ("*Kiss*") filed on July 21, 1999, issued on November 19, 2002, with an earliest potential priority date of July 21, 1998, and/or are invalid under 35 U.S.C. § 103 as obvious in view of *Kiss* or in combination with the reference(s) specifically identified in the following claim chart or one or more other references identified in Defendants' Preliminary Invalidity Contentions ("Defendants' Invalidity Contentions").

To the extent a finder of fact determines that the references cited herein do not teach certain limitations in the asserted claims, such limitations would have been inherent and/or obvious. These claims are also invalid as obvious in view of *Kiss* alone or in combination with other prior art references, including, but not limited, to the prior art identified in the Cover Pleading of Defendants' Invalidity Contentions, the prior art described in the claim charts attached in Exhibit C, and/or the prior art identified in Exhibit D.

Defendants' Invalidity Contentions are based, in part, upon Defendants' present understanding of the asserted claims and IPA's apparent interpretations of the asserted claims in its July 10, 2019 Preliminary Infringement Contentions ("Infringement Contentions") and Defendants' investigation to date. Defendants are not adopting IPA's constructions or apparent constructions, nor are Defendants admitting to the accuracy of any particular contention or construction. The citations provided in the charts below are exemplary rather than exhaustive and Defendants reserve the right to rely upon additional references uncovered through further searching, other portions of the cited references and/or other portions of references cited within these Invalidity Contentions. Defendants further incorporate by reference the reservation of rights identified in the cover pleading to these Invalidity Contentions as though fully set forth herein.

[1] *Kiss* claims priority to provisional application 60/093,522 which was filed on July 21, 1998, before the January 5, 1999 earliest possible filling date of the '560 Patent.

	Claim Language	Invalidity in View of Prior Art
1(p)	A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of	To the extent the preamble is held to be limiting, *Kiss* discloses a computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of claim 1. *See* Claim Chart for U.S. Patent No. 6,851,115 in view of Kiss, Ex. A-15 ("'115 chart"), claim 1.
1(a)	a plurality of service-providing electronic agents;	*Kiss* discloses a plurality of service-providing electronic agents. *See* '115 chart, claim 1.
1(b)	a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:	*Kiss* discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,:* The meta agent layer analyzes user queries or problem formulations from the user interface layer, allocates tasks to the knowledge agent layer, resolves conflicts arising from the knowledge agent layer, and consolidates (including fusing and deconflicting) results provided by the knowledge agent layer. The knowledge agent layer provides an interaction mechanism for knowledge modules having associated knowledge agents within the knowledge agent layer. Each agent in the system includes inter-agent abstract communications facilities with the capability to negotiate with each other, conduct joint planning, and to collaborate in the execution of planned tasks. In addition to the three layers just mentioned, an agent service layer provides services for maintaining a registry of agents in the system, as well as supporting the distributed problem solving. The registry identifies each agent's capabilities and interests, and contains knowledge about the relationships between them. The meta agent layer and the knowledge agent layer may confer with the agent service layer to identify those other resources capable of furthering the problem-solving

	Claim Language	Invalidity in View of Prior Art
		process. A matchmaking facility is provided for notifying agents interested in a capability of other agents that provide the capability. *Kiss* at 3:25-47. *See also* Fig. 1, Fig. 10, Fig. 5, Figs. 8-21, 2:61-67, 3:33-36, 5:20-64, 5:65, 8:61-64, 12:21-14:30.
1(c)	an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment; and	*Kiss* discloses an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment. *See* '115 chart, claim 1(a).
1(d)	a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves:	*Kiss* discloses a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves. *See* '115 chart, claims 1(e), (h).
1(e)	using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal; and	*Kiss* discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* '115 chart, claim 1(g)-(i). *See also, e.g.*, [user's request of] "what is the effect of increasing sales by 20%?" *Kiss* at 12:24. The system then generates and performs the following sub-goal requests in response to this base request.

	Claim Language	Invalidity in View of Prior Art
		[sales agent asking] "what is the market price at that number of units?" *Kiss* at 12:54-56. [sales agent asking] "the meta agent 707 to confirm that the cost per unit at the specified number of units does not exceed the market price plus an acceptable profit." *Kiss* at 13:25-27. [production agent asking] "the meta agent 707 identify the cost of a sufficient number of production lines to produce the specified number of units." *Kiss* at 13:37-39. [production agent asking] "the meta agent 707 to identify the material cost of a particular material." *Kiss* at 13:56-57. [production agent asking] "the meta agent 707 to identify the labor cost of a number of workers sufficient to support the production lines," *Kiss* at 14:3-5. One advantage of the present invention over existing technologies is that inferencing is distributed and cooperative over a distributed environment. In other words, the problem-solving process has been removed from a centrally-located reasoning mechanism and made granular. Rather than relying on a single knowledge-based system to formulate and execute a problem-solving process, inferencing mechanisms are distributed to many, smaller knowledge systems with each having a more clearly defined set of interests and products. Each smaller knowledge system is provided with knowledge processing capabilities for its domain of knowledge. A meta agent is responsible for decomposing a general inquiry into a series of constituent tasks. Each task is formulated based on knowledge of the capabilities of the underlying knowledge systems. By

	Claim Language	Invalidity in View of Prior Art
		cooperating with each other, the meta agent and knowledge agents at each knowledge system accomplish each task toward solving the global problem. *Kiss* at 3:48-65. *See also* 3:61-62, 5:24-30, 7:20-8:4, 8:32-48, 11:15-16, 12:1-17, 13:25-27,
1(f)	wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes:	*Kiss* discloses wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes. *See* '115 chart, claims 1(a)-(c).
1(g)	a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events.	*Kiss* discloses a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events. *See* '115 chart, claims 1(b)-(c).
20	A computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process.	*Kiss* discloses a computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process. *See, e.g.,:* As illustrated in FIG. 21, the knowledge management system 100 may be interconnected via available network services, such as the Internet, with other, similar systems to form a large scale, global system. The layered architecture of a user interface layer 105, a meta agent layer 107, and a knowledge agent layer 109 supports such scalability. *Kiss* at 14:30-36.

EXHIBIT J

Exhibit C-9

Invalidity of U.S. Patent No. 7,069,560 ("'560 Patent")
by MECCA

As shown in the claim chart below, the asserted claims of the '560 patent are invalid under 35 U.S.C. § 102(a) and/or (b)[1] as anticipated by Andreas Lux, et al., "Understanding Cooperation: an Agent's Perspective" ("MECCA")[2], and/or are invalid under 35 U.S.C. § 103 as obvious in view of MECCA, or in combination with the reference(s) specifically identified in the following claim chart or one or more other references identified in Defendants' Preliminary Invalidity Contentions ("Defendants' Invalidity Contentions").

To the extent a finder of fact determines that the references cited herein do not teach certain limitations in the asserted claims, such limitations would have been inherent and/or obvious. These claims are also invalid as obvious in view of MECCA alone or in combination with other prior art references, including, but not limited, to the prior art identified in the Cover Pleading of Defendants' Invalidity Contentions, the prior art described in the claim charts attached in Appendix C, and/or the prior art identified in Appendix D.

Defendants' Invalidity Contentions are based, in part, upon Defendants' present understanding of the asserted claims and IPA's apparent interpretations of the asserted claims in its July 10, 2019 Preliminary Infringement Contentions ("Infringement Contentions") and Defendants' investigation to date. Defendants are not adopting IPA's constructions or apparent constructions, nor are Defendants admitting to the accuracy of any particular contention or construction. The citations provided in the charts below are exemplary rather than exhaustive and Defendants reserve the right to rely upon additional references uncovered through further searching, other portions of the cited references and/or other portions of references cited within these Invalidity Contentions. Defendants further incorporate by reference the reservation of rights identified in the cover pleading to these Invalidity Contentions as though fully set forth herein.

[1] MECCA was published in 1995, more than one year before the January 5, 1999 earliest possible filing date of the '560 patent.
[2] Published by the Association for the Advancement of Artificial Intelligence, from the Proceedings of the First International Conference on Multiagent Systems (1995).

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(p)	A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of	To the extent that the preamble is held to be limiting, MECCA discloses a computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of claim 1. *See* Claim Chart for U.S. Patent No. 6,851,115 in view of MECCA, Ex. A-9 (“’115 chart”), claim 1.
1(a)	a plurality of service-providing electronic agents;	MECCA discloses a plurality of service-providing electronic agents. *See* ’115 chart, claim 1.
1(b)	a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:	MECCA discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,* MECCA at 264. “In the extended process view of cooperating agents, local goals of an agent become *shared goals* if they are solved in cooperation with other agents. The common planning phase leads to the development of so-called *multi-agent plans,* plans which are executed by more than one agent. During the planning and scheduling process, a multi-agent plan is subdivided - as far as possible - into several single-agent plans which can be executed by the agents. The execution of single-agent tasks does not only comprise head or body actions but also communicator actions (i.e. sending cooperation primitives) to coordinate the behavior of other agents.”

	’560 Patent Claim Language	Invalidity in View of Prior Art

		Speaker	Hearer
PROPOSE(g)	Pre:	$g \in \mathcal{G}$	-
	Eff:	$\oplus$ *proposed*(g, H)	$\oplus$ *has_goal*(S, g)
ACCEPT(g)	Pre:	*has_goal*$(H, g) \wedge g \in \mathcal{G}$	*proposed*$(g, S) \wedge g \in \mathcal{G}$
	Eff:	-	$\oplus$ *has_goal*(S, g)
REJECT(g)	Pre:	*has_goal*$(H, g) \wedge g \notin \mathcal{G}$	*proposed*$(g, S) \wedge g \in \mathcal{G}$
	Eff:	$\ominus$ *has_goal*(H, g)	-
MODIFY/REFINE(g, g')	Pre:	*has_goal*$(H, g) \wedge g \notin \mathcal{G} \wedge g' \in \mathcal{G}$	*proposed*$(g, S) \wedge g \in \mathcal{G}$
	Eff:	*proposed*(g', H)	$\oplus$ *has_goal*(S, g')

Figure 1: Cooperation Primitives for Goal Activation

		Speaker	Hearer
PROPOSE($\mathcal{P}_g$)	Pre:	$\mathcal{P}_g \in \mathcal{P}$	-
	Eff:	$\oplus$ *prop_hyp_plan*$(\mathcal{P}_g, H)$	$\oplus$ *has_hyp_plan*$(S, \mathcal{P}_g)$
ACCEPT($\mathcal{P}_g$)	Pre:	*has_hyp_plan*$(H, \mathcal{P}_g) \wedge \mathcal{P}_g \in \mathcal{P}$	*prop_hyp_plan*$(\mathcal{P}_g, S) \wedge \mathcal{P}_g \in \mathcal{P}$
	Eff:	-	$\oplus$ *has_hyp_plan*$(S, \mathcal{P}_g)$
REJECT($\mathcal{P}_g$)	Pre:	*has_hyp_plan*$(H, \mathcal{P}_g) \wedge \mathcal{P}_g \notin \mathcal{P}$	*prop_hyp_plan*$(\mathcal{P}_g, S) \wedge \mathcal{P}_g \in \mathcal{P}$
	Eff:	$\ominus$ *has_hyp_plan*$(H, \mathcal{P}_g)$	-
MODIFY/REFINE$(\mathcal{P}_g, \mathcal{P}'_g)$	Pre:	*has_hyp_plan*$(H, \mathcal{P}_g) \wedge \mathcal{P}_g \notin \mathcal{P} \wedge \mathcal{P}'_g \in \mathcal{P}_g$	*prop_hyp_plan*$(\mathcal{P}_g, S) \wedge \mathcal{P}_g \in \mathcal{P}$
	Eff:	*prop_hyp_plan*$(\mathcal{P}'_g, H)$	$\oplus$ *has_hyp_plan*$(S, \mathcal{P}'_g)$

Figure 2: Cooperation Primitives for Planning

		Speaker	Hearer
TELL(res)	Pre:	$res \in \mathcal{W}_c \wedge$ *goal*(*knows*(H, res))	-
	Eff:	$\oplus$ *knows*(H, res)	$\oplus$ *knows*$(S, res) \wedge res \in \mathcal{W}_c$[15]

Figure 3: Cooperation Primitives for Execution

See also, MECCA at 263.

> “The formal agent model serves as the basis for the description of the cooperation model. However, agents do not always plan and act alone in a world, but must often cooperate with each other to commonly achieve their goals. Cooperation arises as several agents plan and execute their actions in a coordinated way. In MECCA not only single-agent behavior but also co-operation is seen from a goal-based viewpoint. The basic elements of

	’560 Patent Claim Language	Invalidity in View of Prior Art
		cooperation are the so-called *cooperation primitives* (Lux, Bomaxius & Steiner 1992). They are a combination of *cooperation types* and *cooperation objects,* which are either a goal, a plan, a task or unspecific information such as results, parameters, or other knowledge. Cooperation primitives axe basic agent head functions, describing communication among agents with a specific intention. They are represented as plans, whose preconditions and effects fix the semantics/intention of the primitives and whose plan procedures consist of a call to the head function handling the communication (head-communicator-interface).” *See also,* MECCA at 265. “Planning. During the planning phase an agent creates all hypothetical plans Pg for a selected goal g thereby keeping them consistent with already existing hypothetical plans P. The agent can find plans which are incomplete. These partial plans have gaps which are conceptually represented in the event-based representation by ‘‘abstract events". Finding appropriate sub-plans, i.e. refinement of an ‘‘abstract event" into a set of events, can be done in cooperation with other agents. A second kind of plans which are treated in a cooperation with other agents are those which contain "foreign events”. "Foreign events" are events the agent can not execute on its own, but which it knows other agents can possibly execute. See Figure 2.” *See also,* MECCA at 265. “Finding appropriate sub-plans, i.e. refinement of an ‘‘abstract event" into a set of events, can be done in cooperation with other agents. A second kind of plans which are treated in a cooperation with other agents are those which contain "foreign events”. "Foreign events" are events the agent can not execute on its own, but which it knows other agents can possibly execute. See Figure 2.”

	’560 Patent Claim Language	Invalidity in View of Prior Art
		MECCA discloses this limitation as identified above. This limitation is also obvious in view of MECCA combined with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X. As Ex. C-X shows, each of these references also discloses this limitation. A person having ordinary skill in the art would have been motivated to combine MECCA with the knowledge of a person having ordinary skill in the art and/or any one or more of the references identified in the corresponding limitation of Ex. C-X rendering this limitation obvious based on one or more of the motivations to combine identified in Section II.A.3.e of the cover pleading to Defendants’ Preliminary Invalidity Contentions and because each of these references relate to the same field of software agent technology and distributed computing environments.
1(c)	an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment; and	MECCA discloses an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment. *See* ’115 chart, claim 1(a).
1(d)	a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves:	MECCA discloses a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves. *See* ’115 chart, claim 1(e), (h).
1(e)	using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative	MECCA discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* ’115 chart, claim 1(g)-(i). *See also,* MECCA at 265 (regarding “using reasoning to determine sub-goal requests

	’560 Patent Claim Language	Invalidity in View of Prior Art
	completion of the base goal; and	based on non-syntactic decomposition of the base goal”). “Planning. During the planning phase an agent creates all hypothetical plans Pg for a selected goal g thereby keeping them consistent with already existing hypothetical plans P. The agent can find plans which are incomplete. These partial plans have gaps which are conceptually represented in the event-based representation by ‘‘abstract events". Finding appropriate sub-plans, i.e. refinement of an ‘‘abstract event" into a set of events, can be done in cooperation with other agents. A second kind of plans which are treated in a cooperation with other agents are those which contain "foreign events”. "Foreign events" are events the agent can not execute on its own, but which it knows other agents can possibly execute. See Figure 2.” *See also,* MECCA at 264-265. “Goal Activation In the goal activation phase, a co-operation is instantiated if an agent can not find a local plan for the goal or finds a plan which would involve actions to be carried out by other agents. The agent can also propose a goal to a cooperating agent if it thinks that a cooperative plan is cheaper or more effective than a local plan. The proposal of a goal g is answered by the partner after examining the function new_goal. The situation is as in Figure 1.” *See also,* MECCA at 264. “In the extended process view of cooperating agents, local goals of an agent become *shared goals* if they are solved in cooperation with other agents. The common planning phase leads to the development of so-called *multi-agent plans,* plans which are executed by more than one agent. During the planning and scheduling process, a multi-agent plan is subdivided - as far as possible - into several single-agent plans which can be executed by the agents. The execution of single-agent tasks does not only comprise head or body actions

	’560 Patent Claim Language	Invalidity in View of Prior Art
		but also communicator actions (i.e. sending cooperation primitives) to coordinate the behavior of other agents.” *See also,* MECCA at 263. “The formal agent model serves as the basis for the description of the cooperation model. However, agents do not always plan and act alone in a world, but must often cooperate with each other to commonly achieve their goals. Cooperation arises as several agents plan and execute their actions in a coordinated way. In MECCA not only single-agent behavior but also co-operation is seen from a goal-based viewpoint. The basic elements of cooperation are the so-called *cooperation primitives* (Lux, Bomaxius & Steiner 1992). They are a combination of *cooperation types* and *cooperation objects,* which are either a goal, a plan, a task or unspecific information such as results, parameters, or other knowledge. Cooperation primitives axe basic agent head functions, describing communication among agents with a specific intention. They are represented as plans, whose preconditions and effects fix the semantics/intention of the primitives and whose plan procedures consist of a call to the head function handling the communication (head-communicator-interface).” *See also,* MECCA at 265. “Finding appropriate sub-plans, i.e. refinement of an ‘‘abstract event" into a set of events, can be done in cooperation with other agents. A second kind of plans which are treated in a cooperation with other agents are those which contain "foreign events”. "Foreign events" are events the agent can not execute on its own, but which it knows other agents can possibly execute. See Figure 2.” *See also,* MECCA at 262.

	’560 Patent Claim Language	Invalidity in View of Prior Art

of a temporal knowledge representation. Hypothetical reasoning is done by means of abduction (Shanahan 1989); event calculus (Kowalski & Sergot 1986; Eshghi 1988) is used for knowledge representation. Thus, a plan p is a set of events. An event e of a plan p is defined by an instantiated action, a partial ordering, the executing agent and a set of constraints on necessary resources.[5]

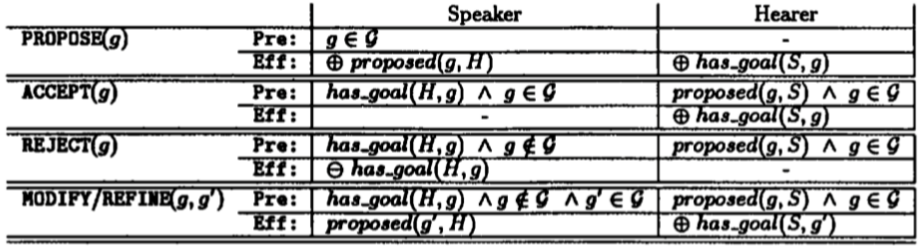
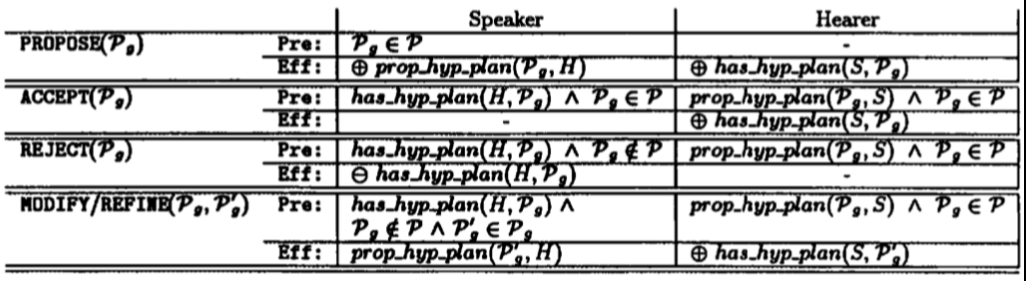
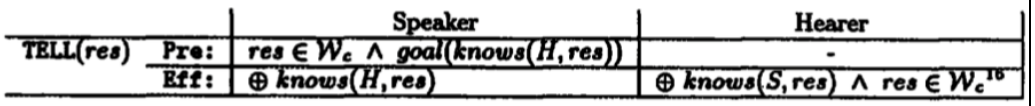
$$e = (a_e, ag_e, V_e, N_e, s_e, ((r_1, C_1), \ldots, (r_n, C_n)))$$

with
a_e = action to be executed
ag_e = executing agent
V_e = set of direct predecessor events
N_e = set of direct successor events
s_e = status of the event,
$s_e \in \{hyp, committed, done\}$
r_i = i-th resource
C_i = set of constraints on the i-th resource, possibly empty

The agent function $plan\_goal : \{\mathcal{G}\} \times \mathcal{W} \rightarrow \mathcal{G}$ selects a goal $g \in \mathcal{G}$ from the set of active goals. The function $find\_all\_plans : \mathcal{G} \times \{\mathcal{A}\} \times \{\mathcal{S}\} \times \mathcal{W} \rightarrow \mathcal{P}$ returns a set of hypothetical plans $\mathcal{P}_g$ which will achieve the goal $g \in \mathcal{G}$, from the world state W_c, ensuring that the plans are compatible with the already scheduled events from $\mathcal{S}$. Knowledge about the post- and preconditions of actions is contained in A. The planning phase can be described by:

```
while (g:= plan_goal(G, W_c)) do
if P_g := find_all_plans(g, A, S, W_c))
then P := P_g ∪ P;
fi;
od.
```

See also, MECCA, Figs. 1-3.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 Figure 1: Cooperation Primitives for Goal Activation  Figure 2: Cooperation Primitives for Planning  **Figure 3: Cooperation Primitives for Execution**
1(f)	wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes:	MECCA discloses wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes. *See* ’115 chart, claims 1(a)-(c).

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(g)	a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events.	MECCA discloses a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events. *See* ’115 chart, claims 1(b)-(c).
20	A computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process.	MECCA discloses a computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process. *See, e.g.,* MECCA at 265. “Planning. During the planning phase an agent creates all hypothetical plans Pg for a selected goal g thereby keeping them consistent with already existing hypothetical plans P. The agent can find plans which are incomplete. These partial plans have gaps which are conceptually represented in the event-based representation by ‘‘abstract events". Finding appropriate sub-plans, i.e. refinement of an ‘‘abstract event" into a set of events, can be done in cooperation with other agents. A second kind of plans which are treated in a cooperation with other agents are those which contain "foreign events”. "Foreign events" are events the agent can not execute on its own, but which it knows other agents can possibly execute. See Figure 2.” *See also,* MECCA at 264. “In the extended process view of cooperating agents, local goals of an agent become *shared goals* if they are solved in cooperation with other agents. The common planning phase leads to the development of so-called *multi-agent plans,* plans which are executed by more than one agent. During the planning and scheduling process, a multi-agent plan is subdivided - as far as possible -

EXHIBIT K

Exhibit C-X

Invalidity of U.S. Patent No. 7,069,560 ("'560 Patent")
by Secondary References

As shown in the claim chart below, the asserted claims of the '560 patent are invalid under 35 U.S.C. § 102(a), 35 U.S.C. § 102(b), and/or 35 U.S.C. § 102(e) as anticipated and/or are invalid under 35 U.S.C. § 103 as obvious, or in combination with the reference(s) specifically identified in the following claim chart or one or more other references identified in Defendants' Preliminary Invalidity Contentions ("Defendants' Invalidity Contentions.").

The Secondary References relied upon herein include, but are not limited to:

- Bian, C., An Experimental Environment for Cooperative Agents, Thesis for the Master's Degree in Computer Science, Department of Computer Science Institute of Mathematical and Physical Sciences University of Tromso (June 30, 1997) ("Bian").
- Bian et al., "ViSe2 – An Agent-Based Expert Consulting System with Efficient Cooperation," Proceedings of the 1997 IEEE International Conference on Intelligent Engineering Systems (Sept. 15-17, 1997) ("Bian 2").
- Cheyer et al., "Multi-Modal Maps Using an Open Agent Architecture," Video, (1995) ("SFMAP").
- Farley, J., Java Distributed Computing, 1st ed. (1998) ("Farley"),
- Georgeff, M., Communication and Interaction in Multi-Agent Planning, AAAI-83 Proceedings (1983) ("Georgeff").
- Malone et al., "Agents for Information Sharing and Coordination: A History and Some Reflections," chapter in Bradshaw, J. (ed.), Software Agents, 1997 ("Malone").
- Moran, et al, "Multimodal User Interfaces in the Open Agent Architecture" ("*Moran reference*") published on January 6, 1997.
- Odubiyi, J. et al., SAIRE - A Scalable Agent-Based Information Retrieval Engine, Proceedings of the First International Conference on Autonomous Agents (Jan. 1997) ("Odubiyi").
- Schwartz et al., Cooperating Heterogenous Systems (1995) ("Schwartz95").
- U.S. Patent No. 6,088,689 to Kohn et al., filed November 29, 1995 and issued July 11, 2000 ("Kohn").
- U.S. Patent No. 5,706,406 to Pollock, filed May 22, 1995 and issued January 6, 1998 ("Pollock").
- U.S. Patent No. 6,029,174 to Sprenger et al., filed October 31, 1998 and issued February 22, 2000 ("Sprenger").
- U.S. Patent No. 6,704,765 to Chang et al., filed December 14, 1994 and issued March 9, 2004 ("Chang").
- U.S. Patent No. 6,065,062 to Periasamy et al., filed on December 10, 1997 and issued on May 16, 2000 ("Periasamy").
- U.S. Patent No. 6,144,992 to Turpin et al., filed on May 9, 1997 and issued on November 7, 2000 ("Turpin").
- U.S. Patent No. 7,028,312 to Merrick et al., filed March 23, 1999 and issued April 11, 2006, with an earliest priority date of March 23, 1998 ("Merrick").

- Open Agent Architecture software and associated documents ("OAA") (Ex. C-1 incorporated by reference herein), including but not limited to:
    - Adam Cheyer, "agentlib.c" (updated Nov. 10, 1996), *available at* http://www.ai.sri.com/~oaa/distribution/distribv1/download/pc/oaa_c.zip (oaa_c\agentlib\agentlib.c) ("OAA Agentlib.c").
    - "Documentation for Open Agent Architecture Agents" (July 19, 1995 to September 20, 1996), *available at* http://www.ai.sri.com/~oaa/distribution/agents/agents.html ("OAA Agents").
    - "OAA Tutorial" (1994-1998), *available at* http://www.ai.sri.com/~oaa/distribution/distribv1/tutorial.html ("OAA Tutorial").
- The Open Agent ArchitectureTM – Building Communities of Distributed Software Agents presentation by Cheyer et al. published at least by February 21, 1998, available at http://www.ai.sri.com/~oaa/oaaslides/ ("Feb 1998 OAA Presentation") (Ex. C-2 incorporated by reference herein).
- OAA PAAM 1998 Presentation published at least by March 23, 1998 and disclosed prior to January 5, 1998 ("PAAM '98 Tutorial") (Ex. C-3 incorporated by reference herein).
- Genesereth, "An Agent-Based Framework of Interoperability" (1997) ("Genesereth '97") (Ex. C-7 incorporated by reference herein).
- Labrou, et al., "Semantics for an Agent Communication Language" (1996) ("Labrou") (Exs. C-7 and C-8 incorporated by reference herein).
- Singh, et al, "A Distributed and Autonomous Knowledge Sharing Approach to Software Interoperation" (March 22, 1995) ("Singh") (Ex. C-8 incorporated by reference herein).
- Lux, et al., "Understanding Cooperation: an Agent's Perspective" ("MECCA") (Ex. C-9 incorporated by reference herein).
- Cheyer, et al., "Multimodal Maps: An Agent-Based Approach" (June 9, 1995) ("Multimodal Maps Paper") (Ex. C-11 incorporated by reference herein).
- Finin, "Software Agents Knowledge Sharing KQML, KIF and Ontologies" (October 16, 1997), available at https://www.csee.umbc.edu/~finin/sisce97/ ("Finin I") (Ex. C-14 incorporated by reference herein).
- U.S. Patent No. 6,484,155 ("*Kiss*") filed on July 21, 1999, issued on November 19, 2002, with an earliest potential priority date of July 21, 1998 (Ex. C-15 incorporated by reference herein).
- "FIPA 97 Specification" by the Foundation for Intelligent Physical Agents ("*FIPA 97*") published on October 10, 1997 (Ex. C-16 incorporated by reference herein).
- Cheyer, et al, MVIEWS: Multimodal Tools for the Video Analyst" ("Cheyer") published on January 6, 1998 (Ex. C-17 incorporated by reference herein).
- Cohen, et al, "An Open Agent Architecture" ("Cohen") published on March 23, 1994 (Ex. C-18 incorporated by reference herein).

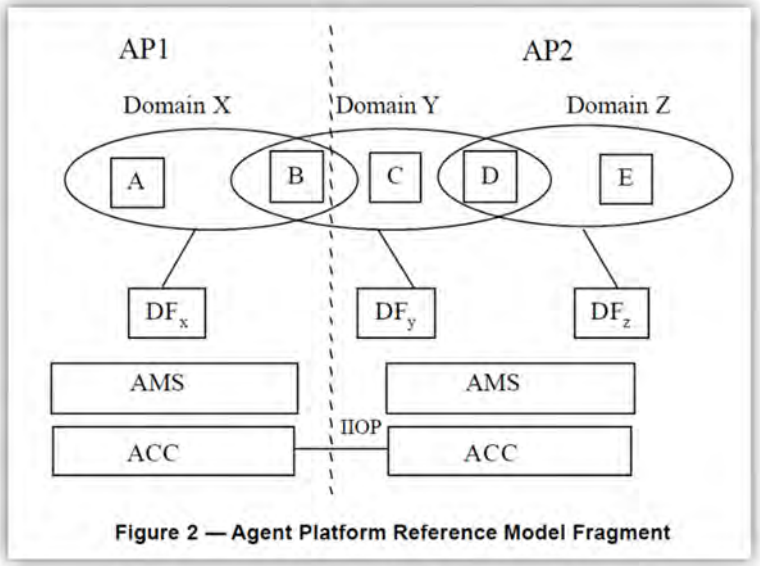
- Martin et al., "Building Distributed Software Systems with the Open Agent Architecture," Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, March 23-25, 1998, London, UK (PAAM98) ("*Martin*") (Ex. C-19 incorporated by reference herein).
- InfoSleuth system and associated documents ("InfoSleuth") (Ex. C-20 incorporated by reference herein), including but not limited to:
    - Nodine et al., "Facilitating Open Communication in Agent Systems: The InfoSleuth Infrastructure," published in Intelligent Agents IV: Agent Theories, Architectures, and Languages, Proceedings 4th International Workshop, ATAL'97 (1998) ("Nodine").
    - Nodine et al., "Experience with the InfoSleuth Agent Architecture" published in the Proceedings of the Fifteenth National Conference on AI, September 27, 1998 ("Nodine 2").
    - Bayardo et al., "InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments," published in the Proceedings ACM SIGMOD International Conference on Management of Data, Vol. 26, Issue 2 (June 1997) ("Bayardo").
    - Urban et al., "Expressing Composite Events in InfoSleuth" published December 1998 ("Urban").
- General Magic system and associated documents ("General Magic") (Ex. C-21 incorporated by reference herein).
- RETSINA system and associated documents ("RETSINA") (Ex. C-22 incorporated by reference herein).
- Finin et al., "KQML as an Agent Communication Language," chapter in Bradshaw, J. (ed.), Software Agents, 1997 ("Finin II") (Ex. C-23 incorporated by reference herein).

To the extent a finder of fact determines that the references cited herein do not teach certain limitations in the asserted claims, such limitations would have been inherent and/or obvious. These claims are also invalid as obvious alone or in combination with other prior art references, including, but not limited, to the prior art identified in the Cover Pleading of Defendants' Invalidity Contentions, the prior art described in the claim charts attached in Appendix C, and/or the prior art identified in Appendix D.

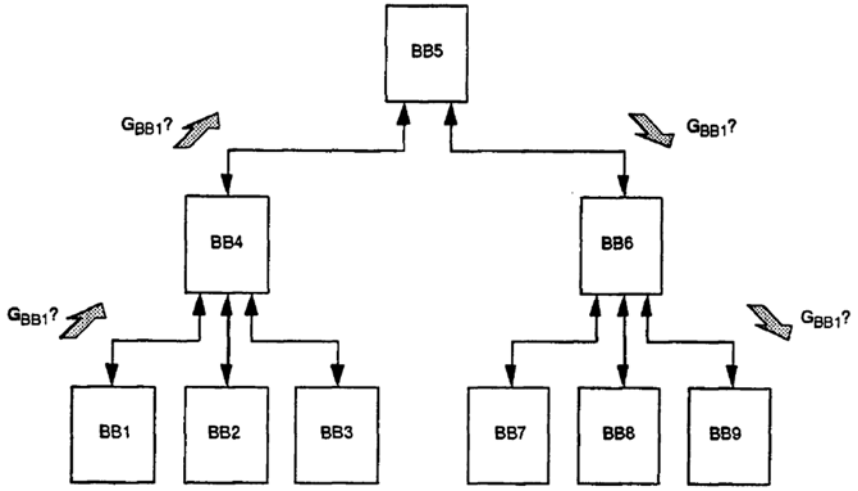
Defendants' Invalidity Contentions are based, in part, upon Defendants' present understanding of the asserted claims and IPA's apparent interpretations of the asserted claims in its July 10, 2019 Preliminary Infringement Contentions ("Infringement Contentions") and Defendants' investigation to date. Defendants are not adopting IPA's constructions or apparent constructions, nor are Defendants admitting to the accuracy of any particular contention or construction. The citations provided in the charts below are exemplary rather than exhaustive and Defendants reserve the right to rely upon additional references uncovered through further searching, other portions of the cited references and/or other portions of references cited within these Invalidity Contentions. Defendants further incorporate by reference the reservation of rights identified in the cover pleading to these Invalidity Contentions as though fully set forth herein.

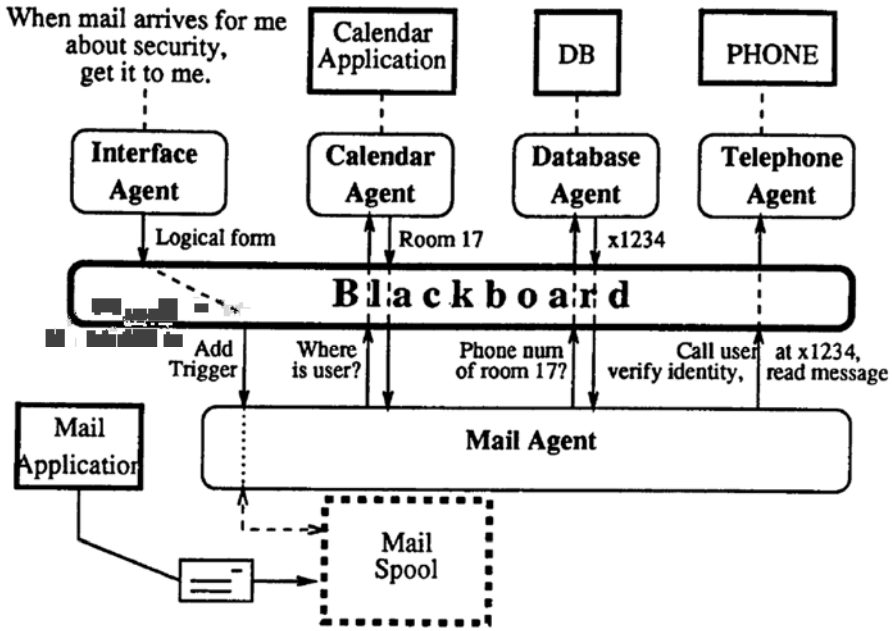
	’560 Patent Claim Language	Invalidity in View of Prior Art
1(p)	A computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of	To the extent that the preamble is held to be limiting, the Secondary References disclose a computer-implemented method for communication and cooperative task completion among a plurality of distributed electronic agents, comprising the acts of claim 1. *See* Claim Chart for U.S. Patent No. 6,851,115 in view of Secondary References, Ex. A-X (“’115 chart”), claim 1.
1(a)	a plurality of service-providing electronic agents;	The Secondary References disclose a plurality of service-providing electronic agents. *See* ’115 chart, claim 1.
1(b)	a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including:	The Secondary References disclose a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See* ’115 chart, claim 1(a); *see also, e.g.,:* *Kiss* discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,:* > The meta agent layer analyzes user queries or problem formulations from the user interface layer, allocates tasks to the knowledge agent layer, resolves conflicts arising from the knowledge agent layer, and consolidates (including fusing and deconflicting) results provided by the knowledge agent layer. The knowledge agent layer provides an interaction mechanism for knowledge modules having associated knowledge agents within the knowledge agent layer. Each agent in the system includes inter-agent abstract communications facilities with the capability to negotiate with each other, conduct joint planning, and to collaborate in the execution of planned tasks.

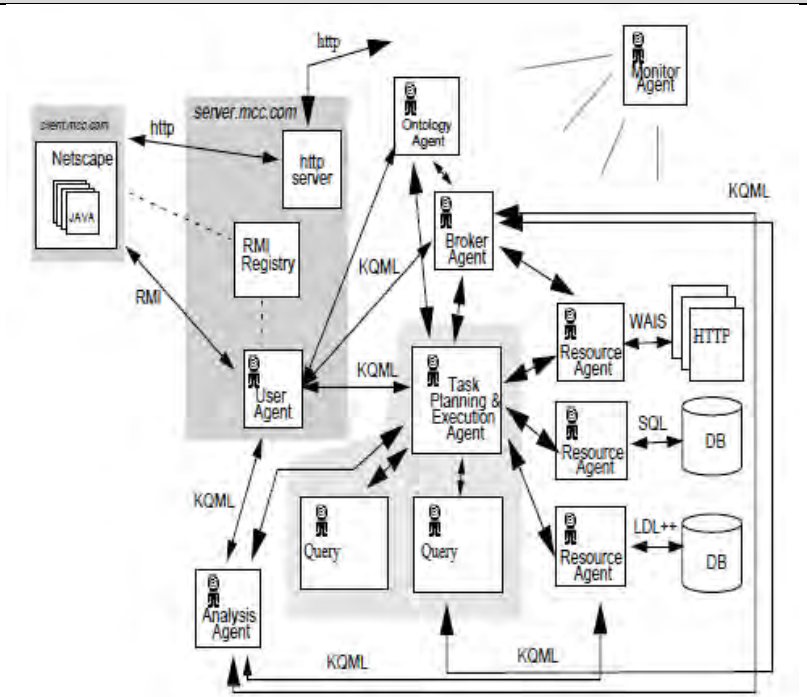
	’560 Patent Claim Language	Invalidity in View of Prior Art
		In addition to the three layers just mentioned, an agent service layer provides services for maintaining a registry of agents in the system, as well as supporting the distributed problem solving. The registry identifies each agent's capabilities and interests, and contains knowledge about the relationships between them. The meta agent layer and the knowledge agent layer may confer with the agent service layer to identify those other resources capable of furthering the problem-solving process. A matchmaking facility is provided for notifying agents interested in a capability of other agents that provide the capability. *Kiss* at 3:25-47. *See also* Fig. 1, Fig. 10, Fig. 5, Figs. 8-21, 2:61-67, 3:33-36, 5:20-64, 5:65, 8:61-64, 12:21-14:30. *FIPA 97* discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,:*

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 Figure 2 — Agent Platform Reference Model Fragment *FIPA 97* at (FIPA97 Part 1): 10. *See also* (FIPA97 Part 1): 6-7. *Cohen* discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g., Cohen* at 2. As discussed above, the Open Agent Architecture contains one blackboard "server" process, and many client agents; client agents are permitted to execute on different host machines. We are investigating an architecture in which a server may itself be a client in a hierarchy of servers; if none of its client agents can solve a particular goal, this goal may be passed further along in the hierarchy. Following Gelerntner's LINDA model [8], blackboard systems

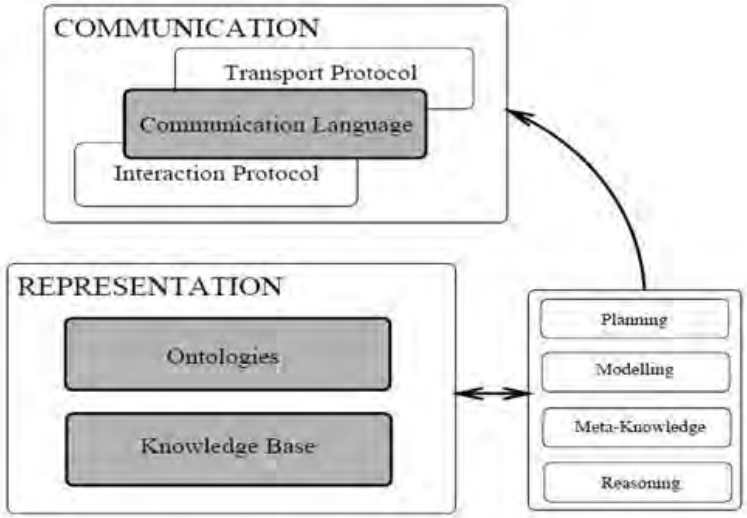
	’560 Patent Claim Language	Invalidity in View of Prior Art
		themselves can be structured in a hierarchy, which could be distributed over a network (see Figure 1).1 When a goal (G) is requested to be posted on a local blackboard (BBi), and the blackboard server agent at BBl determines that none of its child agents has the requisite capabilities to achieve the goal, it propagates the goal to a more senior blackboard server agent (BB4) in the hierarchy. BB4 maintains a knowledge base of the predicates that its lower level blackboards can evaluate. When a senior server receives such a request, it in turn will propagate the request down to its subsidiary servers. These subsidiary servers either have immediate client agents who can evaluate the goal, or can them selves pass on the goal to another subsidiary server. In the case illustrated in Figure 1, BB4 determines that none of its subsidiary blackboards can handle the goal, and thus sends the goal to its superior agent (BB5). BB5 passes the goal to BB6, who in turn passes it to BB9. When such a referred goal is passed through the hierarchy of blackboards, it is accompanied by information about the originating blackboard (indicated by the BBl subscript on G), including information identifying its input port, host machine, etc. This continuation information will enable a return communication (with answers or failure) to be routed to the originating black board. Also, the identity of the responding knowledge source BB9 can be sent back to the originator, so that future queries of the same type from BB1 may be addressed directly to BB9 without passing through the hierarchy of blackboards. *See also Cohen* at Figure 1.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 Figure 1: Hierarchy of Blackboard Servers *See also Cohen* at Figure 2.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 Figure 2: Example of agent interaction *See also Cohen* at 5-6. Bayardo reference discloses a distributed facilitator agent functionally distributed across at least two computer processes.

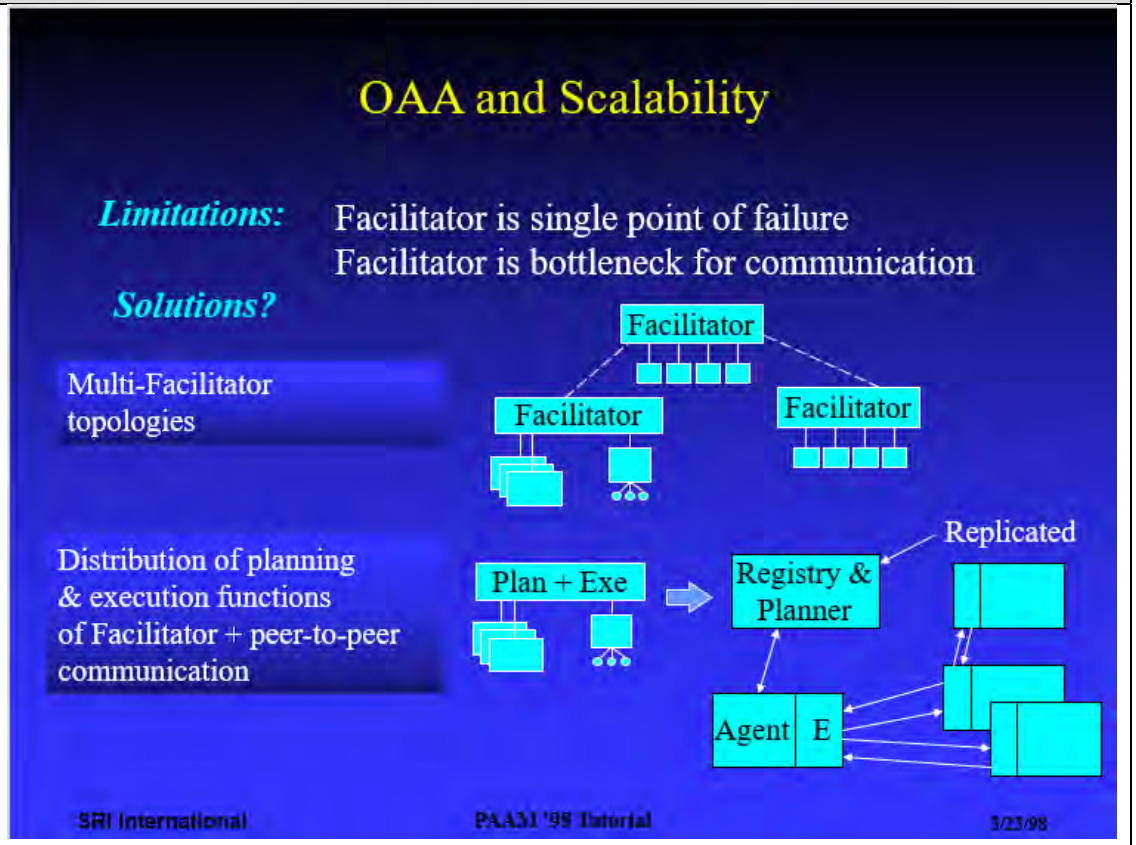
	’560 Patent Claim Language	Invalidity in View of Prior Art
		 Figure 1: The InfoSleuth architecture *See, e.g.*, Bayardo at 195 (FIG. 1). We intend to enhance the brokering capabilities, splitting the broker agent into a family of cooperating, specialized brokers. We will factor out the syntactic brokering capabilities into a separate type of broker agent, possibly implementing it as an ORB interface using CORBA [30]. Semantic brokering will be available at different levels—for example, local to the site, local to the enterprise, and between enterprises. Semantic brokering may include additional information on contents, and additional semantic information such as quality and cost of information.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		See also Bayardo at 205. > We are in the process of splitting the current execution agent into two separate agents, a query decomposition agent and a task execution agent. The task execution agent will develop execution plans based on user requirements using generative planning and plan retrieval utilizing case-based reasoning techniques [17, 31]. The task execution agent may interleave planing with information-gathering subtasks [2, 34, 23] and repair plans when unexpected situations are encountered [10, 26]. Plans will be specified as (transactional) work flows that can be executed by InfoSleuth. It will supervise the execution of the resulting work flows, including managing the transactions it generates. The query decomposition agent will be called by the task execution agent when it has a query over multiple resource agents. It will optimize and decompose queries over multiple resource agents, reassemble the results, and return them to the task execution agent. Bayardo at 205 > After a query is formulated in terms of the selected common ontology, it is sent to the task execution agent that best meets the user's needs with respect to the current query context. Bayardo at 197. Finin II discloses a distributed facilitator agent functionally distributed across at least two computer processes. *See, e.g.*, Farley at Figure 1:

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 Figure 1: Our abstract model for interoperating software agents identifies three classes of components – representation components, communication components, and components not directly related to shared understanding. Farley reference discloses an agent functionally distributed across multiple processes. *See, e.g.*, Farley at 4 (“Agents can be distributed across multiple processes . . . . There are two objects running in distinct processes on separate machines, but together we can consider them to make up one customer agent . . . .”). > For the sake of this book, we will use the term “agent” as a general way to refer to significant functional elements of a distributed application. . . . Agents can be distributed across multiple processes, and can be made up of multiple objects and threads in these processes. Our customer agent might be made up of an object in a process running on a client desktop that's listening for data and updating the local display, along with an object in a process running on the bank server, issuing queries and sending the data back to the client. There are two objects running in distinct processes on separate machines, but together we can consider them to make up one customer agent, with client-side elements and server-side elements. So a distributed application can be thought of as a

	’560 Patent Claim Language	Invalidity in View of Prior Art
		coordinated group of agents working to accomplish some goal. Each of these agents can be distributed across multiple processes on remote hosts, and can consist of multiple objects or threads of control. *See also* Farley at 3-4. The motivations for distributing an application this way are many. Here are a few of the more common ones: Computing things in parallel by breaking a problem into smaller pieces enables you to solve larger problems without resorting to larger computers. . . . Large data sets are typically difficult to relocate, or easier to control or administer located where they are, so users have to rely on remote data servers to provide needed information. Redundant processing agents on multiple networked computers can be used by systems that need fault tolerance. If a machine or agent process goes down, the job can still carry on. *See also* Farley at Title (“Java Distributed Computing”), Farley at 1 (explaining distributed computing); Farley at 1. PAAM ’98 Tutorial discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See e.g.,* PAAM ’98 Tutorial at 37 (disclosing distributed facilitator architecture).

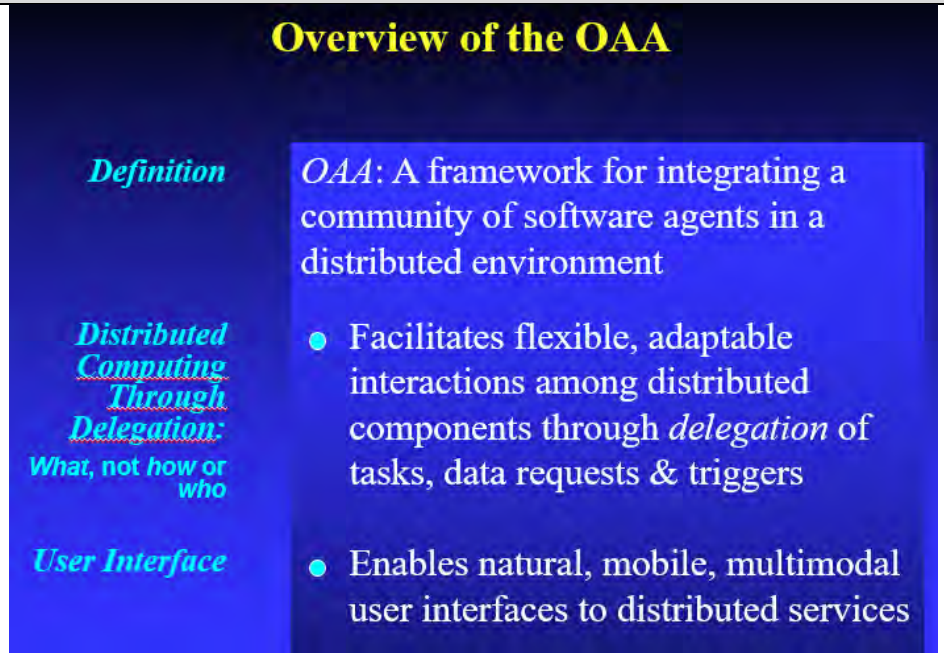
	'560 Patent Claim Language	Invalidity in View of Prior Art

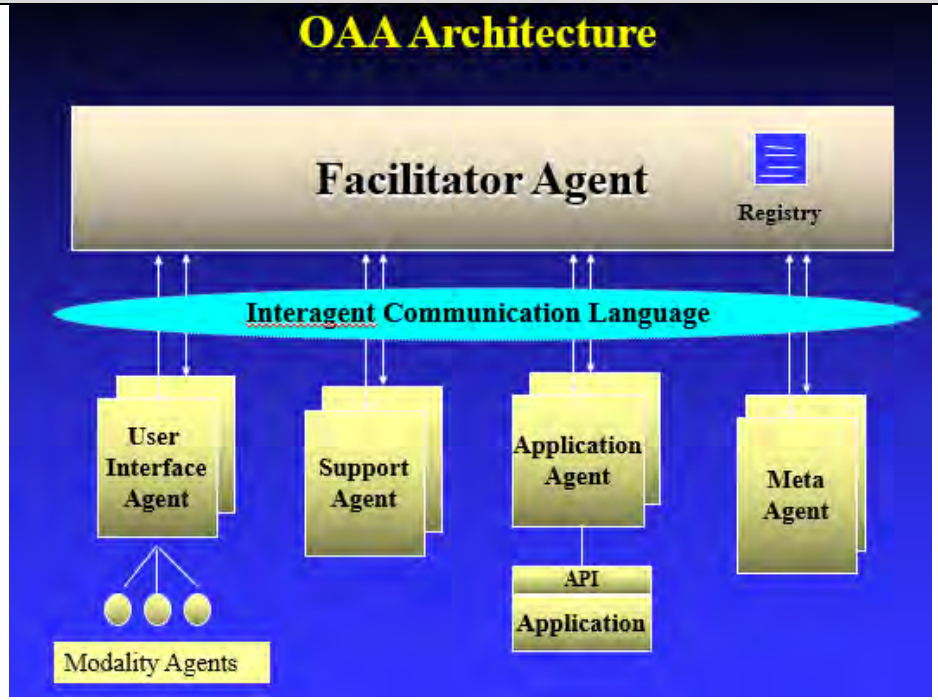


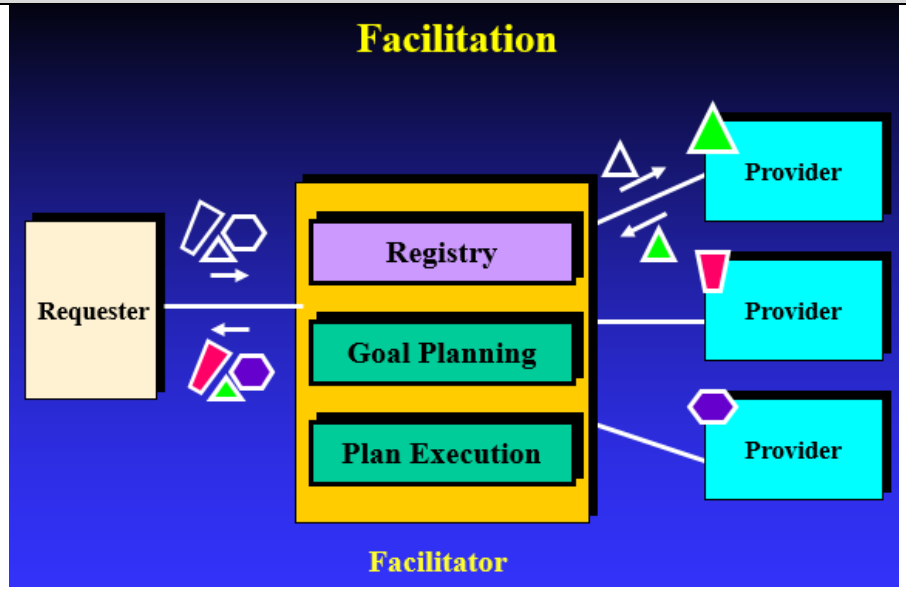
The OAA is currently a research prototype exploring techniques for enabling more flexible interactions among distributed components. As pointed out in this slide, there are certain limitations with respect to the scalability of the simple agent-client, facilitator-server pictures shown previously. Here are two possible strategies for retaining OAA's flexibility while sidestepping some of the scalability issues:

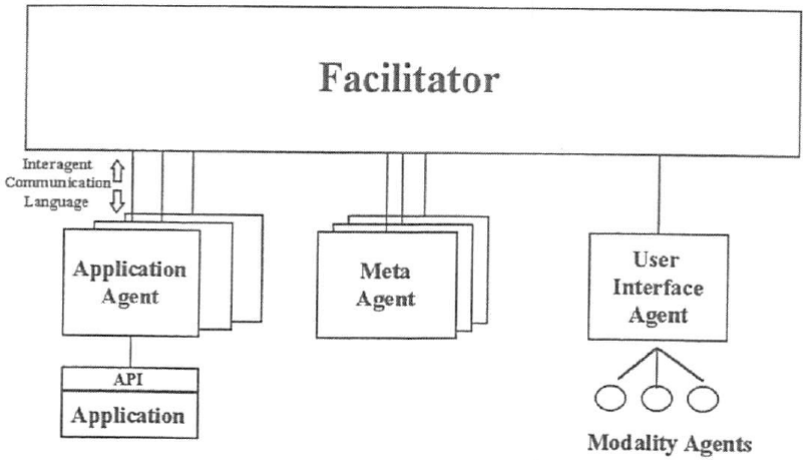
1. First, an OAA Facilitator is an agent like another, so other network topologies of agent configurations can be constructed. One example (but not the only

	’560 Patent Claim Language	Invalidity in View of Prior Art
		possibility) is a hierarchical construction, where a top level Facilitator manages collections of both client agents and other Facilitators. Facilitator agents could be installed for individual users, for a group of users, or as appropriate for the task. 2. A second approach is a bit more difficult to explain, but seems very promising. In the current version of OAA, a Facilitator agent, upon receiving a complex request, will break the request into pieces, discover which agents can participate in the resolution process, and then constructs an execution plan describing who, how and in what order the agents will interact to accomplish the overall goal. Once the plan is constructed, the Facilitator begins executing the plan, sending requests to agents in parallel, collecting and routing answers to other agents, etc. By separating these two phases (planning & execution), the planner and registry component can easily be replicated across many machines, and the execution state of a complex request, instead of the Facilitator storing this for all agent requests (e.g., single point of failure), can be distributed among many agents. In addition, it is clear that techniques such as load-balancing, resource management, and dynamic configuration of agent locations and numbers could be naturally incorporated into the Facilitator, using any of the OAA topologies discussed. *See, e.g.*, PAAM ’98 Tutorial at p.12 (disclosing that an agent’s functionality can be distributed across multiple processes).

	'560 Patent Claim Language	Invalidity in View of Prior Art
		 *See also*, PAAM '98 Tutorial at pp.13, 26 (disclosing that various agents are in bi-directional communication with the facilitator agent) (emphasis in original). *Support agents* provide very general services of great importance to many different systems and/or many different agents within a system; for example, natural language agents translate users' requests into ICL. *Application agents* provide services, which may be domain-independent or domain-dependent. Application Agents are frequently wrapped versions of legacy programs.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 *See also* PAAM ’98 Tutorial at p.31 (disclosing that the communication between the facilitator and various agents are bi-directional).

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 *Martin* discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,*: Second, an OAA facilitator is distinguished by its handling of compound goals, which are introduced above (Section 4.3.1). This involves three types of processing: delegation, that is, completion of the addresses embedded within a compound goal; optimization of the completed goal, including parallelization where appropriate; and interpretation of the optimized goal. *Martin* at 368. *See also, e.g.:* Agent-based systems have shown much promise for flexible, fault-tolerant,

	’560 Patent Claim Language	Invalidity in View of Prior Art
		distributed problem solving. *Id*. at 373. First, there are a variety of multi-facilitator topologies that can be exploited in constructing large systems. It would be useful to investigate which of these exhibits the most desirable properties with respect to both scalability and fault tolerance. Second, it is possible to modularize the facilitator’s key functionalities. For example, goal planning (delegation and optimization) can readily be separated from goal execution. Given this, one can envision a configuration in which the execution task is distributed to other agents, thus freeing up the facilitator. *Martin* at 374.  Figure 1: OAA System Structure. *Id*. at Figure 1. Genesereth ’97 discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional

	’560 Patent Claim Language	Invalidity in View of Prior Art
		communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,* Genesereth ’97 at 320 (bold emphasis added). > Facilitators and the agents they manage are typically organized into what is often called a *federated system.* Figure 2 illustrates the structure of such a system in the simple case in which there are just three machines, one with three agents and two with two agents apiece. As suggested by the diagram, agents do not communicate directly with each other. Instead, they communicate only with their **local facilitators**, and **facilitators, in turn, communicate with each other**. In effect, the agents form a “federation” in which they surrender their autonomy to the facilitators. *See also,* Genesereth ’97 at Fig. 2.

	’560 Patent Claim Language	Invalidity in View of Prior Art

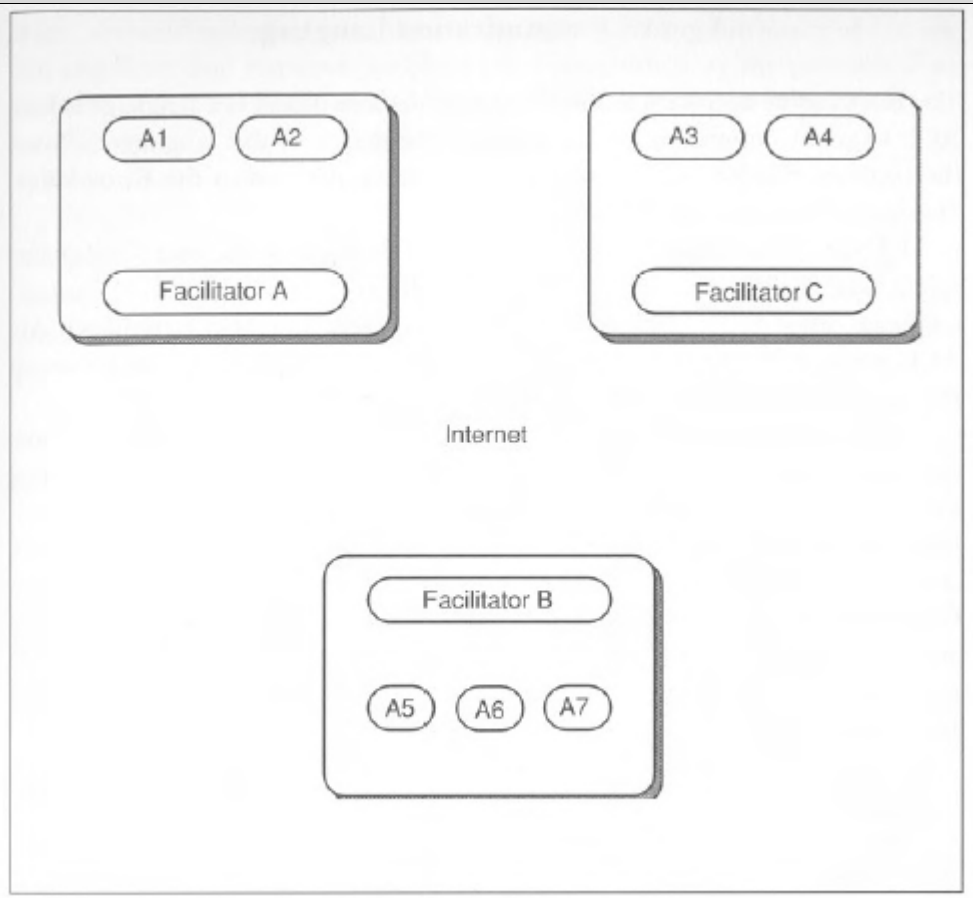


Figure 2. Federated system.

See also, Genesereth ’97 at 320 (emphasis added).

> As with most other brokering approaches, messages from servers to facilitators are undirected; i.e., they have content but no addresses. It is the responsibility of the **facilitators to route such messages to agents able to handle them**. There can be an arbitrary number of facilitators, on one or more machines, and

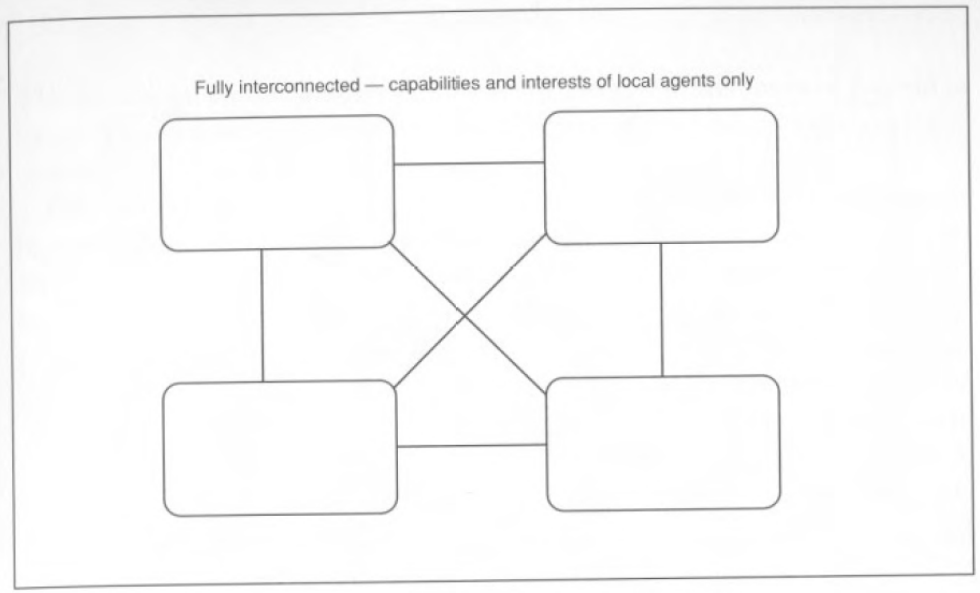
	’560 Patent Claim Language	Invalidity in View of Prior Art
		the **network of facilitators** can be connected arbitrarily. *See also,* Genesereth ’97 at 320 (bold emphasis added). The federation architecture provides assisted coordination of other agents based on a *specification-sharing* approach to interoperation. Agents can dynamically connect or disconnect from a facilitator. **Upon connecting to a facilitator, an agent supplies a specification of its *capabilities* and *needs* in ACL. In addition to this meta-level information, agents also send application-level information and requests to their facilitators and accept application-level information and requests in return. Facilitators used the documentation provided by these agents to transform these application-level messages and route them to the appropriate agents.** The agents agree to service the requests sent by the facilitators, and in return, the facilitators manage the requests posted by the agents. *See also,* Genesereth ’97 at 320-21. A major difference between the knowledge-sharing approach to software interoperation and previous approaches lies in the sophistication of the processing done by these facilitators. In some cases, facilitators may have to translate the messages from the sender's form into a form acceptable to the recipient. In some cases, they may have to decompose the message into several messages, sent to different agents. In some cases, they may combine multiple messages. In some cases, this assistance can be rendered interpretively, with messages going through the facilitators; in other cases, it can be done in one-shot fashion, with the facilitators setting up specialized links between individual agents and then stepping out of the picture. *See also,* Genesereth ’97 at 325-26 (emphasis added). In the approach to interoperation described here, application programmers write their programs as software agents. Like other agents, **a software agent is obliged to communicate in ACL**, but it does so in a particularly stylized way:

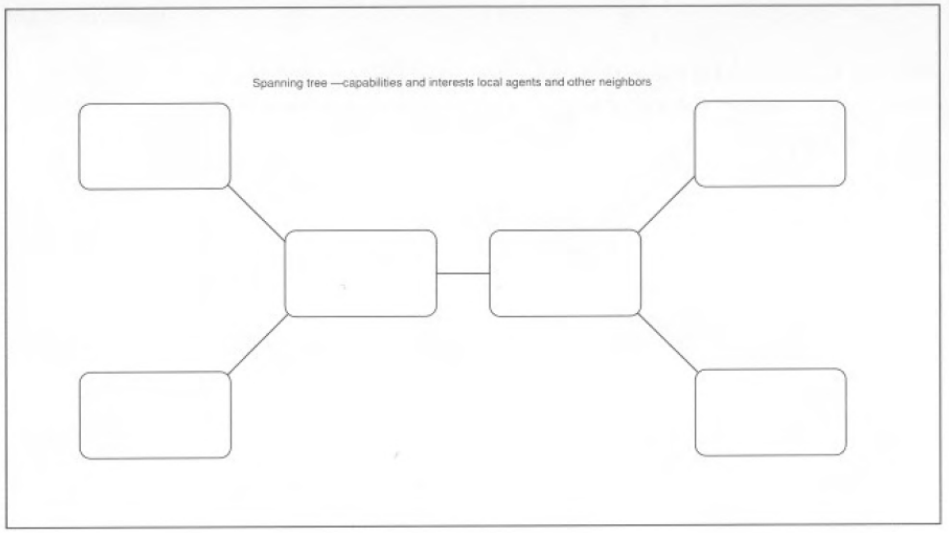
	’560 Patent Claim Language	Invalidity in View of Prior Art
		l. On start up, it initiates an ACL connection to the local facilitator. **2. It supplies the facilitator with a description of its capabilities.** 3. It then enters normal operation: **it sends the facilitator requests when it is incapable of fulfilling its own needs, and it acts to the best of its abilities to satisfy the facilitator's requests**. A software agent is a special kind of agent in that it surrenders its autonomy to the facilitator. A general agent is not compelled to satisfy the requests of other agents. It can accept them or decline them, or it can negotiate for payment. A software agent does not have this freedom. **After registering with its local facilitator and supplying its specification, the software agent is obliged to satisfy the facilitator's requests whenever it can. Of course, this is a good deal in many cases, since the agent gets the facilitator's services in return**. *See also,* Genesereth ’97 at 326 (emphasis added). **Specifying Agent Capabilities and Needs** In order to provide services to other agents, **an agent must communicate its capabilities to the facilitator in ACL. An agent specifies its capabilities by transmitting “handles” facts to its facilitator.** For example, an agent capable of answering questions about the dealer of a vendor may transmit the following specification to its facilitator: (handles business-agent‘(ask-one ,?variables(dealer ,?dealer ,?vendor))) (handles business-agent‘(ask-all ,?variables(dealer ,?dealer ,?vendor))) These facts state that agent business-agent is capable of answering queries about a single dealer for a vendor, or all the dealers for a vendor. The actual capability is a quoted KQML expression, such as ‘(ask-one ,?variables(dealer ,?dealer ,?vendor)) in the first example. This specification is similar to the object interface specifications in CORBA’s IDL.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		If some other agent A_1 wants to know the dealers of NEC, it may communicate the following request to the facilitator: (ask-all ?x (dealer ?x nec)) The facilitator examines its knowledge base and determines that the business-agent can handle the request. **The facilitator sends the request to the business-agent, gets the answer, and passes it to A_1**. Agent A_1 is completely unaware of the sequence of steps performed in servicing its request. *See also,* Genesereth ’97 at 326-27 (emphasis added). **An agent specifies its needs by transmitting “interested” facts to its facilitator.** For example, the following states that the agent cs-manager is interested in all facts regarding the release of PC-compatible computers. (interested cs-manager ‘(tell (released ,?manufacturer PC ,?model))) Similar to “handles” statements, “interested” statements can be conditional: (<= (interested cs-manager ‘(tell (released ,?manufacturer PC ,?model))) (member ?manufacturer ‘(ibm toshiba nec micro-international))) This states that the cs-manager agent is interested only in the release of PC-compatible computers from IBM, Toshiba, NEC, and Micro-International. **If another agent transmits the following fact to the facilitator:** **(tell (released micro-international PC 6500D))** **then the facilitator will examine its knowledge base and find that the agent cs-manager is interested in expressions of this form, and it will send the same KQML expression to the cs-manager**. *See also,* Genesereth ’97 at 329 (emphasis added). Facilitators are the system-provided agents that coordinate the activities of the other agents in the federation architecture. **Each facilitator keeps the other facilitators in the network informed of which agents are connected to it and**

	’560 Patent Claim Language	Invalidity in View of Prior Art
		what facts have been communicated by them. *See also,* Genesereth ’97 at 329 (bold emphasis added). Facilitators provide a collection of services, including: • *White Pages:* finding the identity of agents by name, for example, “What agents are connected?” or “Is agent x connected?” • *Yellow Pages:* finding the identity of agents capable of performing a task. For example, “What agents are capable of answering the query x?” • *Direct Communication:* sending a message to a specific agent. • *Content-based Routing:* the facilitator is given the responsibility of handling a request. It makes use of the specifications and other information provided by the agents to do this, thereby giving the illusion that it is the sole provider of all services. • *Translation:* agents may use different vocabulary. In order to interoperate, the facilitator may have to translate the vocabulary of one agent into the vocabulary of another. • *Problem Decomposition:* handling a complex request may require breaking it into sub-problems, getting the answers to the sub-problems, and then combining these answers to obtain the answer to the original request. As in content-based routing, the facilitator makes use of the specifications and application-specific information provided by the agents to accomplish this. • *Monitoring:* when an agent informs the facilitator of a need, the facilitator monitors its knowledge to determine if the need can be satisfied. For example, an agent may specify the need “I am interested in facts about the position of chips in design x.” **The responsibility of the facilitator on each machine is to assist the agents running on that machine to collaborate with each other and, indirectly, with the agents running on other machines.** *See also,* Genesereth ’97 at 336-38 (disclosing multiple facilitators on same machine each on a different processor and connection of remote and local agents and facilitators)

	’560 Patent Claim Language	Invalidity in View of Prior Art
		(emphasis added). Since remote communication is more expensive than local communication, there is good reason for having at least one facilitator on each machine. Otherwise, in order for a program to communicate with another program on the *same* machine, it would have to send a message to a *remote* machine! On the other hand, there is really no reason to have more than one facilitator per machine. Anything that can be handled by two facilitators can be handled by one facilitator. **There can be no computational advantage, unless the two facilitators are running on different processors with the same machine**. What about the connection of agents to facilitators? While it is possible to consider a situation in which every agent is connected to every facilitator, this is impractical in settings, like the Internet, where there are likely to be many agents and many facilitators. For this reason, in federation architecture, I assume that every agent is connected to one and only one facilitator Finally, there is the issue of inter-facilitator connectivity. Here, there are multiple choices, each with advantages and disadvantages. The simplest sort of architecture is full interconnection, as suggested by figure 4. In this architecture, every facilitator is connected to every other facilitator. Since these connections are logical connections and not physical wires, this sort of architecture is feasible, though not necessarily desirable.

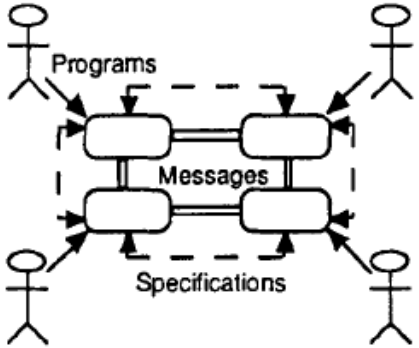
	’560 Patent Claim Language	Invalidity in View of Prior Art
		 *Figure 4. Full interconnection architecture.* * * * An alternative that alleviates this difficulty is a spanning tree architecture, as suggested in figure 5. In this approach, facilitators are connected in such a way that there is a path from every facilitator to every other facilitator but there are no loops.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 *Figure 5. Spanning tree architecture.* * * * Finally, there is the general connectivity architecture. In this architecture, every facilitator is connected at least indirectly with every other facilitator, as in the spanning tree architecture, but there is no restriction that the connectivity be loop free. * * * Unfortunately, it has the disadvantage of possible loops. If one facilitator sends a message to a second and the second passes it on to a third and the third passes it on again, it might end up back where it started. Fortunately, loops of this sort can be caught by adding sender information to each message (as in many mail protocols, for example) and checking for this information when a message is received. It can also be handled by having each

	’560 Patent Claim Language	Invalidity in View of Prior Art
		facilitator save information about which messages it has sent. Either way the loops can be broken. The programming cost is a little higher, but the efficiency and reliability of the approach recommend it highly. Another complexity in the spanning tree and general connectivity architectures stems from the need of facilitators to merge the interests of other facilitators in with those of their own agents in complicated ways. In a full connectivity architecture, each facilitator simply aggregates the interests of its local agents and passes those interests on to all other facilitators. Each facilitator uses this information to handle incoming requests. In the other two architectures, the interests passed on to neighbors are more complicated. A facilitator connected to two other facilitators must blend the interests of its first neighbor into the interests of its local agents in the specification it sends to its second neighbor; and it must blend the interests of its second neighbor into the interests of its local agents in the specification it sends to its first neighbor. *See also,* Genesereth ’97 at 339 (emphasis added). This section presents a simple example of the **federation architecture**. Instead of focusing on the details, I present a broad picture of the types of software interoperation made possible. First, a brief overview of the scenario. There is a computer systems manager in a publishing company who wants to upgrade the computers used by the sales staff to portable Pentium-based machines. The computer systems manager **informs the facilitator of his interest** in Pentium laptops. Sometime later, the **computer product agent notifies the facilitator of the availability** of a Pentium laptop, and this information is passed on to the computer systems manager by the facilitator. The computer systems manager **asks the meeting scheduling agent to set up a joint meeting** with the managers of the sales and finance departments to discuss the purchase of the new machines. **The meeting scheduling agent gets the available times from the calendar agents** for the sales and finance managers to schedule a meeting.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		See also, Genesereth ’97 at 341. Finally, the example also illustrates the dual nature of agents as both providers and consumers of services. For example, the meeting scheduling agent can handle a request to schedule a meeting. However, in order to service this request, the scheduling agent must ask the facilitator for the calendars of the participants. *See also,* Genesereth ’97 at 343-44 (emphasis added). Similarly, it is not possible to put a bound on the total number of agents in the system. **A system can have a network of facilitators, with different agents connected to different facilitators.** Each facilitator must be capable of transmitting a request to any agent that can handle it, independent of its location. To minimize the number of capability and interest specification facts, each facilitator summarizes the capabilities and interests of its directly connected agents, and passes on this summary to its neighboring facilitators. The summary reduces the number of facts and may involve generalization. For example, if one directly-connected agent can answer questions about the dealers of Apple computers and another directly-connected agent can answer questions about IBM dealers, then the facilitator may summarize the answers by informing its neighboring facilitators that it can answer questions about the dealers of all personal computers. There is a space-time tradeoff here: fewer less-precise specifications vs. a larger number of more precise specifications. It is acceptable for an agent to handle a request by indicating that it cannot answer it, for example if its specifications are too general. Singh discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,* Singh at 339 (disclosing that an agent’s functionality can be distributed across multiple processes) (emphasis added). [An] approach [supports] software interoperation based on specification sharing. Software components, called **agents**, provide

	’560 Patent Claim Language	Invalidity in View of Prior Art
		machine processable descriptions of their capabilities and needs. Agents can be realized in different programming languages, and they **can run in different processes on different machines**. In addition, agents can be dynamic - at run time agents can join the system or leave. The system uses the declarative agent specifications to automatically coordinate their interoperation. *See also*, Singh at 341, FIG. 1 (disclosing the agents are in bi-directional communications with facilitators) (emphasis added). Our efforts . . . rely on a highly expressive communication language called ACL (for Agent Communication Language). Programs (called agents) use ACL to supply machine-processable documentation to system programs (called facilitators), which then coordinate their activities. The agent-based approach to **interoperability** is based on the notion of shared abstraction . . . Individual programmers can write their programs without knowledge of the specific vocabulary or interfaces of other software components. Computer users can avail themselves of the **services of different programs** by asking their systems to **coordinate their interaction**. The view of **automated coordination** is illustrated in the right half of Fig. 1 (reproduced below).

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 *See also* Singh at 363-64 (disclosing that the communication between agents and facilitators are bi-directional). . . . Whenever a new piece of information is added to the product agent's knowledge base it notifies the facilitator of it . . . The facilitator performs inference to see if any agent is interested . . . The facilitator cannot answer these questions locally, and it forwards the queries to the product-agent who can answer them . . . The facilitator passes on the two queries of the scheduler to the sm-datebook agent and the fm-synchronize agent. *See also,* Singh at 353.

	'560 Patent Claim Language	Invalidity in View of Prior Art

Figure 5 shows a picture of a Federation Architecture in the simple case where there are three machines, with one facilitator per machine, one machine with three agents, and the remaining with two agents each. Agents are restricted to communicate directly with facilitators. There can be an arbitrary number of facilitators, on one or more machines, and the network of facilitators can be connected arbitrarily.

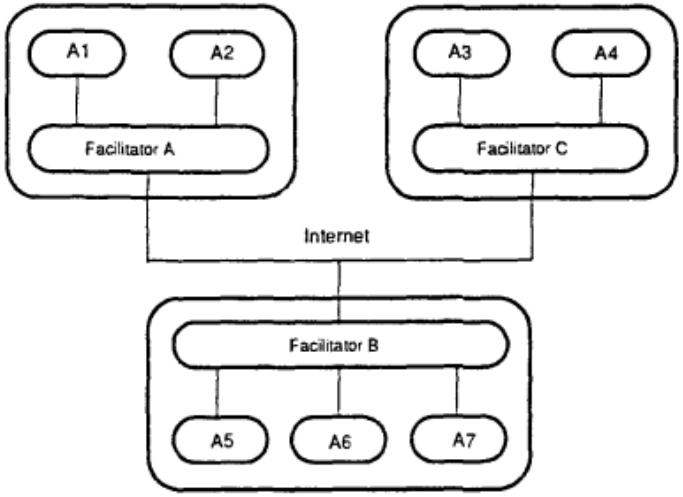


Fig. 5. Federated system.

Odubiyi reference discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including.

See, e.g., Odubiyi reference at 296-297.

3.6 Multi-Agent Coordination Techniques

SAIRE's multi-agent architecture, also shown in Figure 3, depicts how a Coordinator agent provides the AM with the system's services such as other agents' skill base and location. The location of each AM is the agent's socket ID.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		Each rectangular box in the diagram bolds an AM in a multi-agent production system (i.e., a CLIPS process). Each AM controls the activities of a group of command-driven agents under its domain. There is no direct communication between the AMs except through the coordinator. . . . **3.6.2 Disadvantages of a Multi-Agent Coordinator Architecture.** A multi-agent system architecture that employs a coordinator formalism has a centralized control scheme that can result in a single point-of failure. This risk can be mitigated in SAIRE by implementing a backup Coordinator agent. **3.6.3 Strategies for Achieving Multi-Agent Coordination in SAIRE.** Each AM, as well as the Coordinator agent in SAIRE is always alive. The AM continuously executes a loop, checking for newly arriving **t**asks, servicing those tasks, and forwarding results to other agents. Tasks are serviced by their delegation to Specialist agents under the AM. Specialist agents remain idle until delegated a task. They then act on the task, send results to their AM, and wait for the next task to work on. Each agent in SAIRE is modeled as a module. To make multiple modules work together in a single CLIPS environment, module focusing becomes a significant issue. To be in focus means that the module is the center of attention (i.e., CLIPS can only see the structures and knowledge base of that module). An agent environment is then implemented in the following manner. The AM is kept in focus most of the time. This allows it to perform its cycling duties of monitoring for incoming tasks and managing current tasks. Due to the hierarchical organization of the CLIPS environment, the AM can effectively monitor and control the delegation of tasks to the Specialist agents. The Specialist agents are usually out of focus. They remain idle until the AM gives them a task to work on. The Specialist agent will wake up by firing a rule when a request is received causing the agent (i.e., module) to come into focus. This allows CLIPS to use facts asserted into the Specialist agents' knowledge-base.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		The request will then be processed. The results are given to the AM by putting the AM in focus, sending the results to the AM, then taking the AM out of focus. If the Specialist agent is finished, focus is returned to the AM. The implementation is effective in creating two independent agents working to achieve a common goal. The AM is continuously cycling to communicate with the outside world, as well as managing its own environment. Simultaneously, specialist agents wait for work to be delegated by their AM. MECCA discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,* MECCA at 264. “In the extended process view of cooperating agents, local goals of an agent become *shared goals* if they are solved in cooperation with other agents. The common planning phase leads to the development of so-called *multi-agent plans,* plans which are executed by more than one agent. During the planning and scheduling process, a multi-agent plan is subdivided - as far as possible - into several single-agent plans which can be executed by the agents. The execution of single-agent tasks does not only comprise head or body actions but also communicator actions (i.e. sending cooperation primitives) to coordinate the behavior of other agents.”

	’560 Patent Claim Language	Invalidity in View of Prior Art

		Speaker	Hearer
PROPOSE(g)	Pre:	$g \in \mathcal{G}$	-
	Eff:	$\oplus$ *proposed*(g, H)	$\oplus$ *has_goal*(S, g)
ACCEPT(g)	Pre:	*has_goal*$(H, g) \wedge g \in \mathcal{G}$	*proposed*$(g, S) \wedge g \in \mathcal{G}$
	Eff:	-	$\oplus$ *has_goal*(S, g)
REJECT(g)	Pre:	*has_goal*$(H, g) \wedge g \notin \mathcal{G}$	*proposed*$(g, S) \wedge g \in \mathcal{G}$
	Eff:	$\ominus$ *has_goal*(H, g)	-
MODIFY/REFINE(g, g')	Pre:	*has_goal*$(H, g) \wedge g \notin \mathcal{G} \wedge g' \in \mathcal{G}$	*proposed*$(g, S) \wedge g \in \mathcal{G}$
	Eff:	*proposed*(g', H)	$\oplus$ *has_goal*(S, g')

Figure 1: Cooperation Primitives for Goal Activation

		Speaker	Hearer
PROPOSE($\mathcal{P}_g$)	Pre:	$\mathcal{P}_g \in \mathcal{P}$	-
	Eff:	$\oplus$ *prop_hyp_plan*$(\mathcal{P}_g, H)$	$\oplus$ *has_hyp_plan*$(S, \mathcal{P}_g)$
ACCEPT($\mathcal{P}_g$)	Pre:	*has_hyp_plan*$(H, \mathcal{P}_g) \wedge \mathcal{P}_g \in \mathcal{P}$	*prop_hyp_plan*$(\mathcal{P}_g, S) \wedge \mathcal{P}_g \in \mathcal{P}$
	Eff:	-	$\oplus$ *has_hyp_plan*$(S, \mathcal{P}_g)$
REJECT($\mathcal{P}_g$)	Pre:	*has_hyp_plan*$(H, \mathcal{P}_g) \wedge \mathcal{P}_g \notin \mathcal{P}$	*prop_hyp_plan*$(\mathcal{P}_g, S) \wedge \mathcal{P}_g \in \mathcal{P}$
	Eff:	$\ominus$ *has_hyp_plan*$(H, \mathcal{P}_g)$	-
MODIFY/REFINE$(\mathcal{P}_g, \mathcal{P}'_g)$	Pre:	*has_hyp_plan*$(H, \mathcal{P}_g) \wedge \mathcal{P}_g \notin \mathcal{P} \wedge \mathcal{P}'_g \in \mathcal{P}_g$	*prop_hyp_plan*$(\mathcal{P}_g, S) \wedge \mathcal{P}_g \in \mathcal{P}$
	Eff:	*prop_hyp_plan*$(\mathcal{P}'_g, H)$	$\oplus$ *has_hyp_plan*$(S, \mathcal{P}'_g)$

Figure 2: Cooperation Primitives for Planning

		Speaker	Hearer
TELL(res)	Pre:	$res \in \mathcal{W}_c \wedge$ *goal*(*knows*(H, res))	-
	Eff:	$\oplus$ *knows*(H, res)	$\oplus$ *knows*$(S, res) \wedge res \in \mathcal{W}_c$[15]

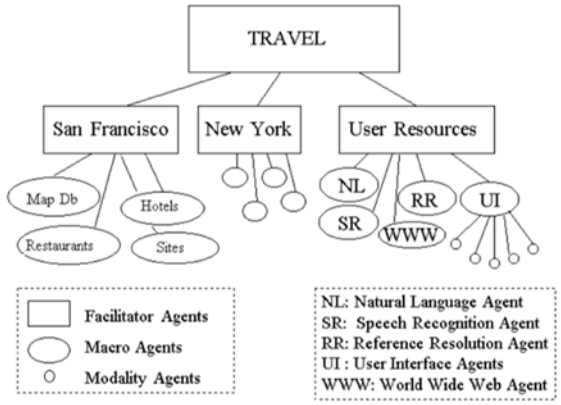
Figure 3: Cooperation Primitives for Execution

See also, MECCA at 263.

> “The formal agent model serves as the basis for the description of the cooperation model. However, agents do not always plan and act alone in a world, but must often cooperate with each other to commonly achieve their goals. Cooperation arises as several agents plan and execute their actions in a coordinated way. In MECCA not only single-agent behavior but also co-operation is seen from a goal-based viewpoint. The basic elements of

	’560 Patent Claim Language	Invalidity in View of Prior Art
		cooperation are the so-called *cooperation primitives* (Lux, Bomaxius & Steiner 1992). They are a combination of *cooperation types* and *cooperation objects,* which are either a goal, a plan, a task or unspecific information such as results, parameters, or other knowledge. Cooperation primitives axe basic agent head functions, describing communication among agents with a specific intention. They are represented as plans, whose preconditions and effects fix the semantics/intention of the primitives and whose plan procedures consist of a call to the head function handling the communication (head-communicator-interface).” *See also,* MECCA at 265. “Planning. During the planning phase an agent creates all hypothetical plans Pg for a selected goal g thereby keeping them consistent with already existing hypothetical plans P. The agent can find plans which are incomplete. These partial plans have gaps which are conceptually represented in the event-based representation by ‘‘abstract events". Finding appropriate sub-plans, i.e. refinement of an ‘‘abstract event" into a set of events, can be done in cooperation with other agents. A second kind of plans which are treated in a cooperation with other agents are those which contain "foreign events”. "Foreign events" are events the agent can not execute on its own, but which it knows other agents can possibly execute. See Figure 2.” *See also,* MECCA at 265. “Finding appropriate sub-plans, i.e. refinement of an ‘‘abstract event" into a set of events, can be done in cooperation with other agents. A second kind of plans which are treated in a cooperation with other agents are those which contain "foreign events”. "Foreign events" are events the agent can not execute on its own, but which it knows other agents can possibly execute. See Figure 2.”

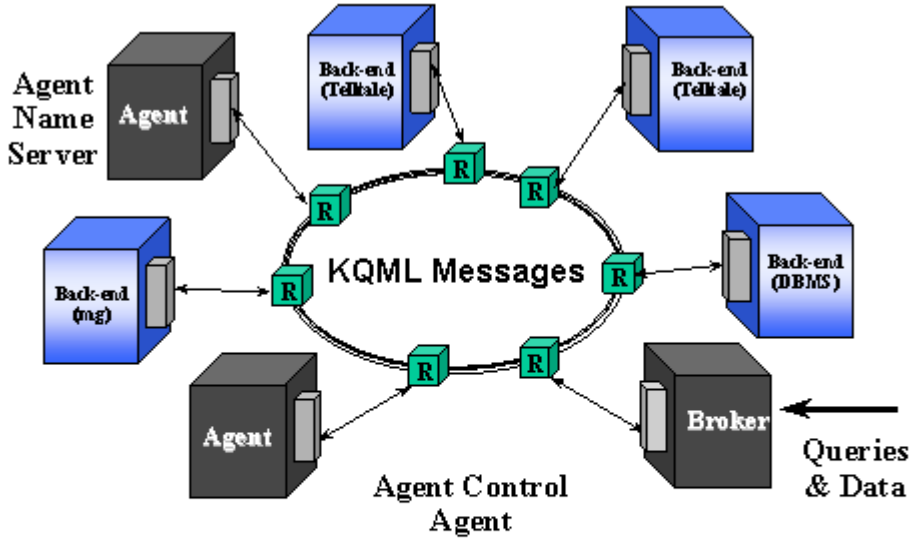
	’560 Patent Claim Language	Invalidity in View of Prior Art
		Multimodal Maps Paper discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,* Multimodal Maps Paper at 7-8. “The Open Agent Architecture (OAA) [5] provides a framework for coordinating a society of agents which interact to solve problems for the user. Through the use of agents, the OAA provides distributed access to commercial applications, such as mail systems, calendar programs database, etc. The Open Agent Architecture possesses several properties which make it a good candidate for our needs: • An Interagent Communication Language (ICL) and Query Protocol have been developed, allowing agents to communicate among themselves. Agents can run on different platforms and be implemented in a variety of programming languages. • Several natural language systems have been integrated into the OAA which convert English into the Interagent Communication Language. In addition, a speech recognition agent has been developed to provide transparent access to the Corona speech recognition system. • The agent architecture has been used to provide natural language and agent access to various heterogeneous data and knowledge sources. • Agent interaction is very fine-grained. The architecture was designed so that a number of agents can work together, when appropriate in parallel, to produce fast responses to queries. The architecture for the OAA, based loosely on Schwartz’s FLiPSiDE system [24], uses a hierarchical configuration where client agents connect to a “facilitator” server. Facilitators provided content-based message routing, global data management, and process coordination for their set of connected agents. Facilitators can, in turn, be connected as clients of other facilitators. Each facilitator records the published functionality of their sub-agents, and when queries arrive in Interagent Communication Language form, they are

	’560 Patent Claim Language	Invalidity in View of Prior Art
		responsible for breaking apart any complex queries and for distributing goals to the appropriate agents. An agent solving a goal may require supporting information and the agent architecture provides numerous means of requesting data from other agents or from the user.” *See also,* Multimodal Maps Paper, Fig. 3.  Figure 3: Agent Architecture for Map Application *SFMAP* discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See e.g.*,;

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 Other agents running on remote machines which could be more powerful process speech and natural language input perform reasoning about context reference and the combination of modalities and access remote databases. The data for the hotels and restaurants in this application are extracted by agents from public worldwide websites on the internet. Users can also access this data through a more conventional web browser if they choose as an integrated part of the system. A final word on some of the advantages of this system. A natural combination of handwriting, speech and gesture may be used when making queries. State-of-the-art natural language and speech recognition systems are integrated using agent technology.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		Secondly, the map interface connects existing databases and knowledge sources including the worldwide web. Thirdly, using the distributed agent architecture allows intensive computing such as speech recognition, natural language processing and database access to be off-loaded to powerful server machines. This interface that you are seeing here can run either on a desktop machine or a handheld pda. And finally, multimedia output including video can be produced if supported by the interface machine. SFMAP at 03:44-5:21. What you are about to see is an unedited demonstration of a synergistic pen voice interface to a distributed set of database which include the worldwide web. This system is implemented on top of the open agent architecture, a framework for allowing a community of software agents to cooperate and communicate in order to accomplish tasks for the user. SFMAP at 00:10-00:30. Finin I discloses a distributed facilitator agent functionally distributed across at least two computer processes, the facilitator agent capable of bi-directional communications with the plurality of service-providing electronic agents, the facilitator agent including. *See, e.g.,* Finin I at 86.

	'560 Patent Claim Language	Invalidity in View of Prior Art
		Some key ideas • Software agents offer a new paradigm for very large scale distributed heterogeneous applications focusing on the interactions of autonomous, cooperating processes which can adapt to humans and other agents. • Intelligence is always a desirable characteristic but is not strictly required by the paradigm. • There is a wealth of prior theory from several disciplines but you needn't to be an expert to to apply or use it. Slide 86 of 87 *See also*, Finin I at 74.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		 Slide 74 of 87
1(c)	an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment; and	The Secondary References disclose an agent registry that declares capabilities for each of the plurality of service-providing electronic agents currently active within the distributed computing environment. *See* ’115 chart, claim 1(a).
1(d)	a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a	The Secondary References disclose a facilitating engine operable to interpret a service request as a base goal, the facilitating engine further operable for generating a goal

	’560 Patent Claim Language	Invalidity in View of Prior Art
	goal satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves:	satisfaction plan associated with the base goal, wherein the goal satisfaction plan involves. *See* ’115 chart, claims 1(e), (h).
1(e)	using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal; and	The Secondary References disclose using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal *See* ’115 chart, claim 1(g)-(i); *see also, e.g.,:* *Kiss* discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* ’115 chart, claim 1(g)-(i). *See also, e.g.,* [user’s request of] “what is the effect of increasing sales by 20%?” *Kiss* at 12:24. The system then generates and performs the following sub-goal requests in response to this base request. [sales agent asking] “what is the market price at that number of units?” *Kiss* at 12:54-56. [sales agent asking] “the meta agent 707 to confirm that the cost per unit at the specified number of units does not exceed the market price plus an acceptable profit.” *Kiss* at 13:25-27. [production agent asking] “the meta agent 707 identify the cost of a sufficient number of production lines to produce the specified number of units.”

	’560 Patent Claim Language	Invalidity in View of Prior Art
		Kiss at 13:37-39. [production agent asking] “the meta agent 707 to identify the material cost of a particular material.” *Kiss* at 13:56-57. [production agent asking] “the meta agent 707 to identify the labor cost of a number of workers sufficient to support the production lines,” *Kiss* at 14:3-5. One advantage of the present invention over existing technologies is that inferencing is distributed and cooperative over a distributed environment. In other words, the problem-solving process has been removed from a centrally-located reasoning mechanism and made granular. Rather than relying on a single knowledge-based system to formulate and execute a problem-solving process, inferencing mechanisms are distributed to many, smaller knowledge systems with each having a more clearly defined set of interests and products. Each smaller knowledge system is provided with knowledge processing capabilities for its domain of knowledge. A meta agent is responsible for decomposing a general inquiry into a series of constituent tasks. Each task is formulated based on knowledge of the capabilities of the underlying knowledge systems. By cooperating with each other, the meta agent and knowledge agents at each knowledge system accomplish each task toward solving the global problem. *Kiss* at 3:48-65. *See also* 3:61-62, 5:24-30, 7:20-8:4, 8:32-48, 11:15-16, 12:1-17, 13:25-27, *FIPA 97* discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* ’115 chart, claim 1(g)-(i).

	’560 Patent Claim Language	Invalidity in View of Prior Art
		Cheyer discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* ’115 chart, claim 1(g)-(i). *See also Cheyer* at 58 (regarding “using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal”). > Instead of preprogrammed unitary method calls to known object services, an agent can express its requests in terms of a high-level logical description of what it wants done, along with optional constraints specifying how the task should be performed. This information is processed by one or more Facilitator agents, which plan, execute and monitor the coordination of the subtasks required to accomplish the end goal. *See also*, *Cheyer* at 58. > The Interagent Communication Language (ICL) provides the means for interaction among agents. When an agent wants to make a request of the agent community, it describes the goal it wants achieved as well as parameters specifying constraints on how the goal is to be accomplished. The request is sent to a Facilitator agent, which uses the declarative specifications it stores about each agent's capabilities, and the parameters defined for the incoming goal, to produce a fully specified execution plan detailing tasks for distributed agents to perform. The Facilitator agent is then responsible for monitoring and coordinating the execution of the plan, by routing requests (potentially to agents in parallel), collecting results, backtracking when subgoals fail, and finally providing the results to the requesting agent. > ICL requests are expressed using the syntax and semantics of Prolog, a decision influenced by our desire to involve the user as closely as possible in agent interactions. ICL expressions can be generated from the Prolog-based logical forms produced by many natural language parsers, allowing the user to make

	’560 Patent Claim Language	Invalidity in View of Prior Art

		requests of the agent community in plain English. As a simple example, the English request "What is the telephone number of John Bear's manager?" would be converted to the ICL expression:
		
		`oaa_Solve( (manager('John Bear', M),`
		`phone_number(M, P)),`
		`[query(var(P))] ).`
		
		See also Cheyer at 57.
		
		 
		
		Figure 4. MVIEWS architecture.
		
		See also Cheyer at 55, 56-58.
		
		Martin discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal.
		
		See, e.g.,:
		
		> Not only does this give the facilitator more information about the nature of a request, but it also makes it possible for the facilitator to decompose compound requests, and delegate the subrequests individually.
		
		Martin at 363.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		See also, e.g.,: Completing the addressing of a goal involves the selection of one or more agents to handle each of its subgoals (that is, each subgoal for which this selection has not been specified by the requester) *Id.* at 368. *See also, e.g.*,: *Martin* discloses that “the facilitator possesses a number of domain-independent coordination strategies” and performs a “multi-agent coordination phase.” *Martin* at 361. *See also, e.g.*,: Completing the addressing of a goal involves the selection of one or more agents to handle each of its subgoals (that is, each subgoal for which this selection has not been specified by the requester). In doing this, the facilitator . . . may [] use strategies or advice specified by the requester. . . .” *Id.* at 368. *See, e.g.*,: [M]eta-agents augment [the facilitator’s strategies] using application-specific knowledge or reasoning (e.g. rules, learning algorithms, planning, and so forth). *Id.* at 361. The facilitator is a specialized server agent that is responsible for coordinating agent communications and cooperative problem-solving. *Id.* at 359, 361.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		Cooperation among the agents of an OAA system is achieved via messages expressed in a common language, ICL, and is normally structured around a 3-part approach: providers of services register capabilities specifications with a facilitator; requesters of services construct goals and relay them to a facilitator, and facilitators coordinate the efforts of the appropriate service providers in satisfying these goals. *Martin* at 362. First, it encompasses a very general notion of *transparent delegation.* This means that a requesting agent can generate a request, and a facilitator can manage the satisfaction of that request, without the requester needing to have any knowledge of the identities or locations of the satisfying agents. In some cases, such as when the request is a data query, the requesting agent may also be oblivious to the *number* of agents involved in satisfying a request. Transparent delegation is possible because agents’ capabilities (solvables) are treated as an abstract description of a service, rather than as an entry point into a library or body of code. Second, an OAA facilitator is distinguished by its handling of compound goals, which are introduced above (Section 4.3.1). This involves three types of processing: *delegation*, that is, completion of the addresses embedded within a compound goal; *optimization* of the completed goal, including parallelization where appropriate; and *interpretation* of the optimized goal. The *delegation* step results in a goal that is unambiguous as to its meaning and as to the agents that will participate in satisfying it. Completing the addressing of a goal involves the selection of one or more agents to handle each of its subgoals (that is, each subgoal for which this selection has not been specified by the requester). In doing this, the facilitator uses its knowledge of the capabilities of its client agents (and possibly of other facilitators, in a multi-facilitator system). It may also use strategies or advice specified by the requester, as explained below.”) (emphasis in original). *Martin* at 368-69.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		See also, e.g.: Distinguishing features of OAA as compared with related work include extreme flexibility in using facilitator-based delegation of complex goals . . . . *Id.* at 355. *See also, e.g.*: User Delegation. Users want to be able to delegate a task to the agent community without having to specify how and who will perform each subpiece of every command. As an example from the Automated Office application, the request “When mail arrives for me from David, get it to me immediately” produces coordinated activity among 15 independent agents to achieve the goal. *Id.* at 359. *See also, e.g.:* While it is possible to specify one or more agents to handle a call (and there are situations in which this is desirable), in general it is advantageous to leave this delegation task to the facilitator. *Id*. at 366. *See also, e.g.:* OAA has been designed so as to encourage developers to employ the paradigm of community, and to minimize their development effort in doing so, by taking advantage of the facilitator’s provision of transparent delegation and handling of compound goals. *Id.* at 369. When the agent is needed, the facilitator sends it a request expressed in the Interagent Communication Language (ICL). The agent parses this request, processes it, and returns answers or status reports to the facilitator.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		Id. at 362. A request for one of an agent’s services normally arrives in the form of an event from the agent’s facilitator. This event is then handled by the appropriate handler. *Id.* at 364. Genesereth ’97 discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* ’115 chart, claim 1(g)-(i). *See also, e.g.*, Genesereth ’97 at 329 (bold emphasis added). Facilitators provide a collection of services, including: • *White Pages:* finding the identity of agents by name, for example, “What agents are connected?” or “Is agent x connected?” • *Yellow Pages:* finding the identity of agents capable of performing a task. For example, “What agents are capable of answering the query x?” • *Direct Communication:* sending a message to a specific agent. • *Content-based Routing:* the facilitator is given the responsibility of handling a request. It makes use of the specifications and other information provided by the agents to do this, thereby giving the illusion that it is the sole provider of all services. • *Translation:* agents may use different vocabulary. In order to interoperate, the facilitator may have to translate the vocabulary of one agent into the vocabulary of another. • ***Problem Decomposition:* handling a complex request may require breaking it into sub-problems, getting the answers to the sub-problems, and then combining these answers to obtain the answer to the original request. As in content-based routing, the facilitator makes use of the**

	’560 Patent Claim Language	Invalidity in View of Prior Art
		specifications and application-specific information provided by the agents to accomplish this. • *Monitoring:* when an agent informs the facilitator of a need, the facilitator monitors its knowledge to determine if the need can be satisfied. For example, an agent may specify the need “I am interested in facts about the position of chips in design x.” The responsibility of the facilitator on each machine is to assist the agents running on that machine to collaborate with each other and, indirectly, with the agents running on other machines. *See also,* Genesereth ’97 at 329-30 (emphasis added). In handling a message, the message handler uses an **automated reasoning program on its knowledge base of specification information**. Our reasoning program is a variation on the method used in Prolog. There are two primary differences. First of all, it handles KIF syntax rather than Prolog syntax. Secondly, unlike Prolog, it is sound and complete for full first-order predicate calculus: it is based on the model elimination rule of inference, the unification algorithm does an occurcheck, the restriction to Horn clauses is removed, and the search is done in iterative deepening fashion. *See also,* Genesereth ’97 at 330-31. Consider a database with the sentences shown below. The predicate p holds of three pairs of objects-a and a, a and b, and b and e. The predicate q is also true of three pairs of objects-a and b, b and e, and e and d. The predicate r is defined to be true of two objects if there is an intermediate object such that p is true of the former object and this intermediate object and q is true of the intermediate object and the latter object. (p a a) (p a b) (p b c)

	’560 Patent Claim Language	Invalidity in View of Prior Art
		(q a b) (q b c) (q c d) (<= (r ?x ?z) (p ?x ?y) (q ?y ?z)) Suppose now, we wanted to know whether r was true of a and c. The trace shown below shows how the reasoning program derives this result. Call: (r a c)? Call: (and (p a ?y| (q ?y c)) Call: (p a ?y) Exit: (p a a) Call: (q a c) Fail: (q a c) Call: (p a ?y) Exit: (p a b) Call: (q b c) Exit: (q b c) Call: (and (p a b) |q b c)) Exit:(r a c) The desired conclusion (r a c) unifies with the conclusion of the last sentence in the knowledge base with the variable ?x bound to a and the variable ?z bound to c. The program thus reduces the original question to the subquestion on the

	’560 Patent Claim Language	Invalidity in View of Prior Art
		second line-in effect the question of whether there is a binding for the variable ?y for which the conjunction is true. In order for this to be true, there must be a binding for ?y for which (p a ?y) is true. The program first finds (p a a) and binds ?y to a. It then tries to prove (q a c). Unfortunately, this fails. So, the program backs up and tries to find another way to satisfy (p a ?y). In so doing, it discovers the fact (p a b) and binds ?y to b. Again it tries to prove (q b c) and in this case succeeds. Since both conjuncts are proved, the conjunction is proved; and, since the conjunction is proved, the original sentence is proved. *See also,* Genesereth ’97 at 331. This program is both sound and complete. In other words, if the program manages to prove a result, then that result must logically follow from the sentences in the database; and if a conclusion logically follows from the database, the method will prove it. Unfortunately, as with all sound and complete reasoning methods for the full first-order predicate calculus, the method does not necessarily terminate. If a conclusion does not follow from the database, the method may spend forever trying to prove it. While this situation does not often arise, it is a real danger for a piece of system code. In order to deal with this difficulty, the facilitator uses a preprocessor to screen sentences before they are added to the facilitator's database. The facilitator adds a sentence if and only if it can prove that doing so will not cause an infinite loop. Note that the problem of making this determination is itself undecidable; so it is not possible to know in all cases whether a sentence will cause an infinite loop. Our facilitator circumvents this difficulty by taking a conservative approach to proving the “safeness” of a set of sentences: it uses a variety of tests to determine whether an inference will terminate. If a database passes the tests, termination is assured. If not, the database may or may not be safe. *See also,* Genesereth ’97 at 332.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		Content-Based Routing From an application programmer’s point of view, communication in a federation architecture is undirected: application programs are free to send messages without specifying destinations for those messages. It is the job of the facilitator to determine appropriate recipients for undirected messages and to forward the messages accordingly. In so doing, the facilitator functions as a broker for the services provided by the servers in its community. In order to see how this is done, consider how the facilitator handles the message shown below. It is being told via one particular encoding that the object named chipl is indeed a computer chip. (tell '(member chip1 chips)) The facilitator is connected to three agents, named layout, domain-editor, and board-editor. These agents have given the facilitator the specification information shown below. (interested layout '(tell (position ,?x ,?r ,?c))) (<= (interested domain-editor '(tell (member ,?x ,?y))) (symbol ?x) (symbol ?y)) (<= (interested board-editor '(tell (= (,?f ,?x) ,?y))) (member ?f (setoff 'row 'col)) (symbol ?x) (natural-number ?y)) In order to determine which agents are interested in this message, the facilitator forms the query (interested ?a '(tell (member chip1 chips))) and uses its reasoning program to find a binding for variable ?a. In this case, there is just

	’560 Patent Claim Language	Invalidity in View of Prior Art
		one, the domain-editor. Consequently, the facilitator sends the message to this agent. (tell '(member chip1 chips)) Note that in making the determination that the domain-editor agent is interested, the facilitator must not only match the pattern in the first line of its specification but also verify properties of the bindings of the variables, in particular that they are both symbols. *See also,* Genesereth ’97 at 333 (emphasis added). As an example of translation, consider a situation in which the facilitator receives the message shown below. As before, it is being told via one particular encoding that the position of a particular chip on a printed circuit board is located in the tenth row and sixteenth column. (tell '(= (pos chip1) (point 10 16))) The facilitator’s agent catalog mentions that an agent named layout is interested in receiving messages of the form (position **), where ** and ** are natural numbers. (<= (interested kb (tell '(position ?x ?m ?n)) (natural-number ?m) (natural-number ?n)) Since the incoming sentence does not have the form specified in this interest, content-based routing alone would not cause any message to be sent to layout. However, let us suppose that the facilitator also has information relating pos and position, as in the following sentence: (<=> (= (pos ?x) (point ?row ?col)) (position ?x ?row ?col)) **Using this sentence together with the sentence (= (pos chip1) (point 10 16)),**

	’560 Patent Claim Language	Invalidity in View of Prior Art
		the facilitator is able to deduce the sentence (position chip1 10 16). In other words, it can translate from one form to the other. It then checks whether any agent is interested in this information, finds layout, and sends the message shown below. (tell '(position chip1 10 16)) *See also,* Genesereth ’97 at 334-35 (emphasis added). The example of translation in the preceding subsection is particularly simple. One incoming message leads to one outgoing message. **In some cases, an incoming message can be handled only by sending multiple messages to multiple agents. In order to handle such messages, the facilitator must be able to synthesize a multi-step communication plan to handle the incoming message**. As an example of this type of message handling, consider the message shown below. As in the last example, the facilitator is being told the position of a particular chip. (tell '(= (pos chip1) (point 10 16))) One difference in this example is that the facilitator’s agent catalog contains the information shown below, documenting an agent interested in row information and col information but not pos information. (<= (interested board-editor (tell '(= (,?f ,?x) ,?y))) (member ?f (setof 'row 'col)) (symbol ?x) (natural-number ?y)) As before, let us assume that the facilitator’s library contains a sentence relating the two vocabularies. (<=> (= (pos ?x) (point ?row ?col))

	’560 Patent Claim Language	Invalidity in View of Prior Art
		(and (= (row ?x) ?row) (= (col ?x) ?col)))
		In this case, there are two conclusions from the original sentence. The facilitator discovers these two conclusions and sends them on to the board-editor agent.
		(tell '(= (row chipl) 100))
		(tell '(= (col chip1) 160))
		Note that if the incoming message had been an ask-if message, the facilitator would have been able to reduce this to two questions: one about the row of the chip and another about the column. In this case, it would first send the row question to board-editor; then, on getting an answer, it would send in the col question; and, on getting that answer, would be able to answer the original question.
		See also, Genesereth ’97 at 339-40.
		The computer systems manager sits at his terminal with a graphical user interface (GUI) and tells the facilitator that he is interested in being told of the availability of PC-compatible Pentium laptops. The GUI commands are translated into the following KIF fact, which is transmitted to the facilitator:
		(<= interested cs-manager '(tell (available ,?manufacturer ,?model-name)))
		(= (denotation ?model-name) ?model); the model from its name
		(computer-family ?model PC)
		(laptop ?model))
		There is a product agent that can answer queries about the computer family a product belongs to (e.g., PC, Apple) and which computers are laptops. It has specified its capabilities by transmitting the following facts to the facilitator:
		(handles product-agent
		'(ask-one ,?variables (computer-family ,?computer ,?family)))

	’560 Patent Claim Language	Invalidity in View of Prior Art
		(handles product-agent '(ask-if (laptop ,?computer))) Whenever a new piece of information is added to the product agent's knowledge base it notifies the facilitator of the fact. A new Micro-International 36000 computer is announced, and information about it is added to the knowledge base of the product agent. The product agent communicates the following KQML message to the facilitator: (tell (available Micro-International 3600D)) The facilitator performs inference to see if any agent is interested in this fact. It finds that the cs-manager agent is interested, but only if the computer family of the 36000 is PC, and if the 36000 is a laptop. The facilitator cannot answer these questions locally. However, it forwards the queries to the product-agent, who can answer them. The product-agent responds positively to both queries, and the cs-manager is notified of the previous availability of the 36000. A message indicating this pops up on the GUI of the computer systems manager. The computer systems manager uses his GUI to ask the facilitator to schedule a one hour meeting with the managers of the sales and finance groups during the week of December 12th to 16th. The GUI transmits the following KQML message to the facilitator: (schedule-meeting (listof sales-manager finance-manager) (interval 12-12-94 12-16-94) 60) There is a scheduling agent that can schedule meetings. It previously transmitted the following fact to the facilitator: (handles scheduler '(schedule-meeting ,?people ,?interval ,?meeting-duration)) The original meeting request is passed on to the scheduler agent by the facilitator. The scheduler is not able to schedule a meeting directly, since it does

	’560 Patent Claim Language	Invalidity in View of Prior Art
		not have access to the calendars of the sales and finance managers. Therefore, the scheduling agent passes on the following query to the facilitator: (ask-one ?x (calendar sales-manager (interval 12-12-94 12-16-94) ?x)) There is a datebook agent for the sales manager that records his calendar. It had previously notified the facilitator of its capability with the following fact: (handles sales-manager-datebook '(ask-one ,?x (calendar sales-manager ,?interval ,?x))) Similarly, there is a synchronize agent that can answer queries regarding the calendar of the finance manager. The facilitator passes on the two queries of the scheduler to the sales-manager-datebook agent and the finance-manager-synchronize agent. The calendars returned by these agents are sent to the scheduling agent, who schedules the earliest possible meeting. The first available meeting time is transmitted to the facilitator, who finally forwards the results to the cs-manager. Labrou discloses using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See also*, Labrou at 1. A crucial component of this paradigm is the communication language, which is the medium through which the attitudes regarding the content of an exchange between software agents are communicated; the communication language suggests whether the content of the communication is an assertion, a request, some form of query *etc.* Knowledge Query and Manipulation Language (KQML) is an agent communication language that consists of primitives (called *performatives*) which allow agents to communicate such attitudes to other agents and find other agents suitable to process their requests. Our research provides semantics for KQML along with a framework for the semantic

	’560 Patent Claim Language	Invalidity in View of Prior Art
		description of KQML-like languages for agent communication. We do so, avoiding commitments to agent models and inter-agent interaction protocols. *See also*, Labrou at 2. This is an example of a KQML message: (ask-if :sender A :receiver B :language prolog :ontology foo :reply-with id1 :content ‘‘bar(a,b)’’) In KQML terminology, *ask-if* is a *performative*. The value of the :content is an expression in some language (in this case in Prolog) or another KQML message and represents the content of the communication (illocutionary) act. The other parameters (*keywords*) introduce values that provide a context for the interpretation of the :content and hold information to facilitate the processing of the message. *See also*, Labrou at 3.1. The following constitutes the semantic description for each of the performatives: **(1)** A natural language description of the performative’s intuitive meaning; **(2)** An expression that describes the content of the communication act and serves as a formalization of the natural language description; **(3)** Preconditions that indicate the necessary state for an agent in order to send a performative (**Pre(A)**) and for the receiver to accept it and successfully process it (**Pre(B)**); **(4)** Postconditions that describe the states of both interlocutors after the *successful* utterance of a performative (by the sender) and after the receipt and processing (but before a counter utterance) of a message (by the receiver). The postconditions (**Post(A)** and **Post(B)**, respectively) hold unless a *sorry* or an *error* is sent as a *response* in order to suggest the unsuccessful processing of the message; **(5)** A completion condition for the performative (**Completion**) that indicates the final state, after possibly a conversation has taken place and the intention suggested by the performative that started the conversation, has been fulfilled; and **(6)** Any comments that we might find suitable to enhance the understanding of the performative.

	’560 Patent Claim Language	Invalidity in View of Prior Art
		See also, Labrou at 3.3. For a KQML message **performative(A,B,X)**, **A** is the :sender, **B** is the :receiver and **X** is the :content of the performative (KQML message). * * * All expressions in our language denote agents’ states. Agents’ states are either actions that have occurred (PROC and SENDMSG) or agents’ mental states (BEL, KNOW, WANT or INT). We allow conjunctions (∧) and disjunctions (∨) of expressions that stand for agents’ states (the resulting expressions represent agents’ states, also), but we do not allow ∧ and ∨ in the scope of KNOW, WANT and INT. *See also*, Labrou at 4.

	'560 Patent Claim Language	Invalidity in View of Prior Art
		We present the semantics for three KQML performatives (*ask-if*, *tell* and *sorry*) in order to illustrate our approach. [1] – **ask-if(A,B,X)** 1. A wants to know what B believes regarding the truth status of the content X. 2. WANT(A,KNOW(A,S)) where S may be any of BEL(B,X), or ¬(BEL(B,X)). 3. **Pre(A)**: WANT(A,KNOW(A,S)) ∧ KNOW(A,INT(B,PROC(B,M))) where M is **ask-if(A,B,X)** **Pre(B)**: INT(B,PROC(B,M)) 4. **Post(A)**: INT(A,KNOW(A,S)) **Post(B)**: KNOW(B,WANT(A,KNOW(A,S))) 5. **Completion**: KNOW(A,S')) where S' is either BEL(B,X) or ¬(BEL(B,X)), but not necessarily the same instantiation of S that appears in $Post(A)$, for example. 6. Not believing something is not necessarily the same with believing its negation (assuming that the language of B provides logical negation), although this may be the case for certain systems. The **Pre(A)** and **Pre(B)** suggest that a proper advertisement is needed to establish them. – **tell(A,B,X)** 1. A states to B that A believes the content to be true. 2. BEL(A,X) 3. **Pre(A)**: BEL(A,X) ∧ KNOW(A,WANT(B,KNOW(B,S))) **Pre(B)**: INT(B,KNOW(B,S)) where S may be any of BEL(B,X), or ¬(BEL(B,X)).

	’560 Patent Claim Language	Invalidity in View of Prior Art
		4. **Post(A)**: KNOW(A,KNOW(B,BEL(A,X))) **Post(B)**: KNOW(B,BEL(A,X)) 5. **Completion**: KNOW(B,BEL(A,X)) 6. The completion condition holds, unless a *sorry* or *error* suggests B’s inability to acknowledge the *tell* properly, as is the case with any other performative. – **sorry(A,B,Id)** 1. A states to B that although it processed the message, it has no response to provide to the KQML message M identified by the `:reply-with` value **Id** (some message identifier). 2. PROC(A,M) 3. **Pre(A)**: PROC(A,M) **Pre(B)**: SENDMSG(B,A,M) 4. **Post(A)**: KNOW(A,KNOW(B,PROC(A,M))) $\wedge$ not$(Post_M(A))$, where $Post_M(A)$ is the **Post(A)** for message M. **Post(B)**: KNOW(B,PROC(A,M)) $\wedge$ not$(Post_M(B))$ 5. **Completion**: KNOW(B,PROC(A,M)) 6. The postconditions for M, as a result of message M do not hold. The not should be taken to mean that the mental state it qualifies should not be inferred to be true as a *result* of this particular message. This does not mean that for example $Post_M(B)$ does not hold if it has already been established by a previous message; it is up to B to decide (perhaps after using additional information) if and how it wants to alter its internal state with respect to the *sorry*. Bian references disclose using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal and using said reasoning to co-ordinate and schedule efforts by the service-providing electronic agents for fulfilling the sub-goal requests in a cooperative completion of the base goal. *See* ’115 chart, claim 1(g)-(i). *See also, e.g.*, Bian 2 reference at 192 (regarding “using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal”).

	’560 Patent Claim Language	Invalidity in View of Prior Art
		The purpose of cooperation is to allow agents to share knowledge and information with each other, and thus to provide the users a virtual access to distributed knowledge or information sources. To satisfy this demand, first of all, an agent must figure out whom can be shared with. Therefore, we let each agent collect, monitor and reason the capabilities of the others and use the captured information to locate a peer agent. An agent's capabilities are actually those abstract descriptions of the goals in its local knowledge sources. *See also* Bian 2 reference at 192 (regarding “using reasoning to determine sub-goal requests based on non-syntactic decomposition of the base goal”) The task-base plays the main role in an on-line cooperation process. When receiving a goal that is beyond its problem solving capabilities, an agent compares the received goal with those task descriptions in its task-base, selects the first matched tuple, and consults the retrieved peer agent for help. In this sense, an agent can directly retrieve the best qualified peer in its task-base, while leaving those evaluation and reasoning work in the agent's free time. *See also* Bian reference at 39-40. **5.3.2 Twin-Base: Cooperator-Base U Task-Base** In a general acquaintance model approach, at run time, when an agent receives a goal that is beyond its capabilities, it evaluates the other agents’ capabilities and skills with the help of its acquaintance models, recognizes those peer agents that have the capabilities to achieve the goal, figures out the most qualified peer, and contacts it for help… The twin-base consists of a cooperator-base and a task-base. The task-base employs the pre-evaluation idea. It provides direct mappings between tasks and the relevant expert agents that can carry out the tasks. In the task-base, each tuple is related to one task and recorded as the follows:

	’560 Patent Claim Language	Invalidity in View of Prior Art
		task_distribution(Task_Description, Agent, Dependence). The Task _Description states the goal that the Agent can perform. It is enhanced from the capability domain. If the goal should be further divided into sub-goals, then domain Dependence specifies those sub-goals and their related agents. If more than one agents can achieve the same goal, tuples containing these agents are sorted in the task-base based on the agents' priorities evaluated as the follows: **Rule 1** Self-solving has the highest priority. If the host agent can carry out the task, then the agent has the highest priority. There are two reasons for us to employ this rule. Firstly, to ask for assistance from the others will increase the communication overhead, and thus reduce the system performance. Secondly, for security reasons, the agent should trust itself more than anyone else. **Rule 2** Non-host agents are sorted corresponding to their state information. It is better to choose the agents with high trust value and good statistics record. The cooperator-base collects stable information of the other agents and acts as an auxiliary base to the task-base. The cooperator-base is built on (1) state information (2) meta-data of other agents' capability. It is recorded as: cooperator(Index, Agent, State). The task-base plays the main role in an on-line cooperation process. Receiving a task that is beyond its problem-solving capabilities, an agent can compare the received task with those Task Descriptions in its task-base, select the first matched tuple, and consult the retrieved peer agent for help. In this sense, an agent can directly retrieve the best qualified peer with the help of its task-base, while leaving those evaluation and reasoning work in the agent's free time. The twin-base modeling is different from other approaches in the sense that it allows agents to perform reasoning work before they accept tasks from their users. As such, the system on-line performance will be increased and the

	’560 Patent Claim Language	Invalidity in View of Prior Art
		repeated reasoning processes can be avoided. With this respect, we argue that the twin-base modeling improves the efficiency of cooperation. In addition, the twin-base modeling is easy for maintenance, and thus the local computation time is acceptable. *See also* Bian reference at 58-59. **6.3.3 Twin-Base** The twin-base plays a key role of the cooperation in the ViSe2 multi-agent system. It keeps the knowledge about how an agent cooperate others. As mentioned, the twin-base consists of a cooperator-base and a task-base. The cooperator-base collects the stable information of cooperative agents in a group, so the knowledge of cooperator-base will simply be fact knowledge. The task-base provides direct mappings between tasks and agents which can solve these tasks. It is defined in three domains as facts, reasoning rules and message rules (Cao 1997). The following is a typical simplified twin-base of a wine agent.

	’560 Patent Claim Language	Invalidity in View of Prior Art

```
COOPERATOR-BASE {
# Information of group members
# address(agent-index, agent-address)
(address(0,self))
(address(1,album.dslab4.1203))
(address(2,travel.diplom5.1211))
(address(4,clothing.diplom0.1200))
}

TASK-BASE-FACTS {
(task_dis(wine_guide,0,1))
(task_dis(music_recommendation,1,3)
}

TASK-BASE-RULES {
# Reasoning Rules
  (actor(Task_Name,Actors):- task_dis(Task_Name,Actor_Index,1),
                             address(Actor_Index,Actors))
  (get_name(Task_Name):- task_dis(Task_Name,0,_))
  (actor_index(Task_Name,Actors,Actor_Index):-
   task_dis(Task_Name,Actor_Index,1), address(Actor_Index,Actors))
  }

  MESSAGE-RULES {
  (({:TYPE add_cooperator}
    {:COOPERATOR Var_Address}
    {:INDEX Var_Index}),
   (1),
   (agt_insert address($Var_Index,[append address_tempp '$Var_Address'])
    agt_agent2agentConn $Var_Address
     ),(),
    (agt_answer "The new cooperator ($Var_Address) is added."))
  }
```

	’560 Patent Claim Language	Invalidity in View of Prior Art
1(f)	wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes:	The Secondary References disclose wherein the plurality of service-providing electronic agents and the distributed facilitator agent communicate using an interagent Communication Language (ICL), wherein the ICL includes. *See* ’115 chart, claims 1(a)-(c).
1(g)	a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events.	The Secondary References disclose a layer of conversational protocol defined by event types and parameter lists associated with one or more of the events, wherein the parameter lists further refine the one or more events. *See* ’115 chart, claims 1(b)-(c).
20	A computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process.	The Secondary References disclose a computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process. *See* ’115 chart, claims 84-85, *See also, e.g.,:* *Kiss* discloses a computer architecture as recited in claim 1 wherein the distributed facilitator agent includes a planning component executing within a first computer process and an execution component executing within a second computer process. *See, e.g.,:* > As illustrated in FIG. 21, the knowledge management system 100 may be interconnected via available network services, such as the Internet, with other, similar systems to form a large scale, global system. The layered architecture of a user interface layer 105, a meta agent layer 107, and a knowledge agent layer 109 supports such scalability. *Kiss* at 14:30-36. *See also* Fig. 1, Fig. 21, 2:61-67, 5:33-36, 5:65.